

REPORT DOCUMENTATION PAGE				Form Approved OMB No. 0704-0188	
Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing the burden, to Department of Defense, Washington Headquarters Services, Directorate for Information Operations and Reports (0704-0188), 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302. Respondents should be aware that notwithstanding any other provision of law, no person shall be subject to any penalty for failing to comply with a collection of information if it does not display a currently valid OMB control number. PLEASE DO NOT RETURN YOUR FORM TO THE ABOVE ADDRESS.					
1. REPORT DATE (DD-MM-YYYY) 14-12-2005		2. REPORT TYPE Final Report		3. DATES COVERED (From – To) 01-Dec-00 - 01-Dec-05	
4. TITLE AND SUBTITLE Mathematical Basis of Knowledge Discovery and Autonomous Intelligent Architectures				5a. CONTRACT NUMBER ISTD Registration No: 1993p	
				5b. GRANT NUMBER	
				5c. PROGRAM ELEMENT NUMBER	
6. AUTHOR(S) Alexander V. Smirnov, Ph.D. Prof.				5d. PROJECT NUMBER	
				5d. TASK NUMBER	
				5e. WORK UNIT NUMBER	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) St. Petersburg Institute For Informatics & Automation of the Russian Academy of Sciences 39, 14th Liniya St. Petersburg 199178 Russia				8. PERFORMING ORGANIZATION REPORT NUMBER N/A	
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) EOARD PSC 802 BOX 14 FPO 09499-0014				10. SPONSOR/MONITOR'S ACRONYM(S)	
				11. SPONSOR/MONITOR'S REPORT NUMBER(S) ISTD 00-7031-2	
12. DISTRIBUTION/AVAILABILITY STATEMENT Approved for public release; distribution is unlimited.					
13. SUPPLEMENTARY NOTES					
14. ABSTRACT Global awareness (GA) entails the acquisition of data from local to global levels, appropriate fusing of the data, and presentation of that data as useful information. This data will then be fused to fully describe situations of interest such as large transportation systems and complex communication systems. This project specifically aims at developing the mathematical basis, architecture and software techniques implementing particular new technologies to support Global Awareness and comprises six main tasks. Task 2 was: 2. Rapid Knowledge Fusion in the Scalable Infosphere;					
15. SUBJECT TERMS EOARD, Mathematical And Computer Sciences, Computer Programming and Software					
16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT UL	18. NUMBER OF PAGES 120	19a. NAME OF RESPONSIBLE PERSON PAUL LOSIEWICZ, Ph. D.
a. REPORT UNCLAS	b. ABSTRACT UNCLAS	c. THIS PAGE UNCLAS			19b. TELEPHONE NUMBER <i>(Include area code)</i> +44 20 7514 4474



**ST. PETERSBURG INSTITUTE
FOR INFORMATICS AND
AUTOMATION**



**EUROPEAN OFFICE OF AEROSPACE
RESEARCH AND DEVELOPMENT
(EOARD)**

Project # 1993P

Mathematical Basis of Knowledge Discovery and Autonomous Intelligent Architectures

Task # 2

Rapid Knowledge Fusion in the Scalable Infosphere

Final Report

Principal Investigator
Head of Computer-Aided Integrated
Systems Laboratory of SPIIRAS

Alexander V. Smirnov, Ph.D. Prof.

**St.Petersburg
November, 2003**

ABBREVIATIONS

AO	application ontology
API	application programming interface
BOM	bill of materials
CGI	common gateway interface
CNP	contract net protocol
COM	component object model
CSP	constraint satisfaction problem
DAML	the DARPA agent markup language
DARPA	Defense Advanced Research Projects Agency
DCOM	distributed component object model
DBMS	database management system
DLL	dynamic link library
DO	domain ontology
DQL	DAML query language
FIPA	Foundation for Intelligent Agents
FTP	file transfer protocol
GSH	Grid service handle
GSR	Grid service reference
HTML	hypertext markup language
HTTP	hypertext transfer protocol
IKB	internal knowledge base
ISAPI	Internet server application programming interface
J2EE	Java™ 2 Enterprise Edition
KB	knowledge base
KF	knowledge fusion
KIF	knowledge interchange format
KL	knowledge logistics
KM	knowledge management
KRS	knowledge representation system
KS	knowledge source
KSNet	knowledge source network
KSO	knowledge source ontology
LDAP	lightweight directory access protocol
OCML	Operational Conceptual Modeling Language
OIL	ontology inference layer

ODBC	open database connectivity
OKBC	open knowledge base connectivity
OL	ontology library
OO	object-oriented
OOCN	object-oriented constraint network
OOTW	operations other than war
PKSO	Preliminary knowledge source ontology
PRO	preliminary request ontology
RDF	resource description framework
RDFS	RDF schema
RO	request ontology
RPC	remote procedure calling
SGML	standard generalized markup language
SOAP	simple object access protocol
SQL	structured query language
TCP/IP	transmission control protocol/Internet protocol
TMO	task and method ontology
UDDI	universal description, discovery, & integration
UML	unified modeling language
UN	United Nations
URI	uniform resource identifier
URL	unified resource locator
VRML	virtual reality markup language
WSDL	Web services description language
WWW	World Wide Web
XML	extensible markup language

CONTENT

1.	Introduction	7
2.	Fusion-Based Knowledge Logistics: Concept and Methodology	9
2.1.	KSNet-Approach for Knowledge Logistics	9
2.2.	Modern Requirements and Standards to Fusion-Based Knowledge Management Systems in the Scalable Infosphere	12
2.2.1.	Modern Requirements to the Fusion-Based Knowledge Management Systems	12
2.2.2.	Open Internet Standards	13
2.2.3.	Web & Grid Services	15
2.3.	Ontology-Driven Methodology of Fusion-Based Knowledge Logistics.....	16
2.3.1.	Scheme of Knowledge Fusion Support	16
2.3.2.	Object-Oriented Constraint Networks as Knowledge Representation Model	17
2.3.3.	Uncertainties in Knowledge Logistics.....	19
2.3.4.	Ontology-Driven Methodology Principles	21
2.3.5.	Knowledge Fusion Patterns.....	22
2.4.	Knowledge Repository Structure	22
2.4.1.	Ontology Library	23
2.4.2.	Knowledge Map	25
2.4.3.	User Profiles	26
2.4.4.	Internal Knowledge Base	27
3.	Multiagent Architecture and Research Prototype of the System “KSNet”	30
3.1.	Agents Community of the System “KSNet”	30
3.2.	Multiagent Architecture.....	34
3.2.1.	Agent Structure.....	34
3.2.2.	Agents’ Interaction.....	36
3.2.3.	Agent Knowledge Representation.....	37
3.3.	Constraint-Based Contract Net Protocol	38
3.3.1.	Conventional Contract Net Protocol	39
3.3.2.	Constraint-Based Negotiation	39
3.3.3.	Modifications of Conventional Contract Net Protocol Interactions	40
3.4.	Implementation of Agents Community	41
3.4.1.	Multiagent Platform and Agent Operations	41
3.4.2.	Conceptual Projects of Agents’ Behavior Scenarios.....	43
3.5.	Structure of the System “KSNet” Research Prototype.....	45
4.	Technologies and Techniques of Problem-Oriented Agents.....	49
4.1.	Configuration Agent: Configuration Management of Knowledge Source Network.....	49
4.2.	Monitoring Agent: Knowledge Indexing and Web Services.....	51
4.2.1.	Knowledge Map Maintenance.....	51
4.2.2.	Web-Service Support	53
4.3.	Knowledge Fusion Agent: On-the-fly Constraint Satisfaction Mechanism	55
4.4.	Translation Agent: Free Text Recognition	60
4.4.1.	Goal Description.....	60
4.4.2.	User Request Parsing.....	60

4.5. Expert Assistant Agent: Knowledge Visualization and Groupware Support for Direct Knowledge Entry	66
4.5.1. Knowledge Visualization by Virtual Reality	66
4.5.2. Expert Collaboration Support.....	66
4.6. Ontology Management Agent: Ontology Library Maintenance.....	68
5. Experiments.....	72
5.1. Experiments on Fusion-based Knowledge Logistics Technology.....	72
5.2. Binni Scenario as a Case Study	73
5.2.1. Health Service Logistics in Coalition Operations Other than War.....	73
5.2.2. General Description of Scenario	75
5.2.3. Binni Scenario for the System KSNet.....	75
5.3. Experimentation on Ontologies.....	78
5.3.1. Application Ontology Creation	78
5.3.2. Verification of Compatibility of Internal Formalism with DAML	84
5.4. Experimentation on Agent Negotiation Protocol	86
5.5. Experimentation on Translation Agent	89
5.5.1. Results of Experiments with Regular Expressions.....	90
5.5.2. Results of Experiments with Stemming	90
5.5.3. Results of Experiments with Spelling	91
5.5.4. Complex Experiments	91
5.6. Experimentation on Knowledge Fusion Agent	92
5.7. Experimentation with Complete Scenario of the System “KSNet”.....	93
5.7.1. Problem Structure and Decomposition.....	93
5.7.2. Knowledge Sources for the Case Study	99
5.7.3. Implementation of Web-Service Interface	100
5.7.4. Experimentation with Different User Preferences.....	102
5.7.5. Experimentation with Uncertainty Factors.....	105
5.8. Agents' Load Analysis.....	109
6. Conclusion and Future Work	111
6.1. Summary of Results	111
6.2. Conclusions Based on Research Prototype Evaluation	112
6.3. Possible Directions of the Project Evolution.....	112
6.4. Publications	114
References.....	117

PREFACE

This document is the Final Report on the task 2 “Rapid Knowledge Fusion in the Scalable Infosphere” of the Project # 1993P “Mathematical Basis of Knowledge Discovery and Autonomous Intelligent Architectures” that is being carried out according to the agreement between European Office of Aerospace Research and Development (EOARD), The International Science and Technology Center (ISTC) and St. Petersburg Institute for Informatics and Automation of the Russian Academy of Sciences (SPIIRAS).

According to the Work Program, at this phase the following tasks have been accomplished: re-design of the models, methods, agent architectures and object-oriented conceptual project of the prototypes knowledge sharing by knowledge maps, and for distributed uncertain knowledge management corresponding to the results of the series of experiments. Integration and testing prototype of an entire agent-based knowledge fusion technology.

This report summarizes the main results of stages B-3.3 of the eleventh and twelfth quarters of the research scheduled by the Work plan. *The developed prototype can be demonstrated by request.*

Principal Investigator

Head of Computer-Aided Integrated Systems Laboratory of SPIIRAS

Alexander V. Smirnov, Ph.D. Prof.

1. INTRODUCTION

Today an intensive knowledge integration and knowledge exchange between participants of the global information environment are required. Along with a large number of distributed knowledge sources representing knowledge in various formats this leads to an appearance of a new direction in knowledge management called knowledge logistics. Knowledge logistics is dealing with activities on acquisition, integration, and transfer of the right knowledge from distributed sources in the right context to the right person in the right time for the right purpose. Knowledge fusion technology was selected as a basis for knowledge logistics. The main aim of the carried out research was to develop methods and tool for rapid knowledge fusion in the distributed information environment to support above activities - to turn distributed knowledge into new knowledge.

The major possible application domains of this research belong to the following areas (Interim Report # 2, 2001):

- Large-scale air operations and/or joint aerospace operations based on large-scale dynamic systems (enterprises) are the distributed operations in an uncertain and rapidly changing environment. Here the information collection, assimilation, integration, interpretation, and dissemination are needed;
- Focused logistics operations and/or Web-enhanced logistics operations addressed sustainment, transportation and end-to-end rapid supply to the final destination. Here the distributed information management and real-time information fusion to support continuous information integration of all participants of the operations are needed;
- Markets via partnerships with industrial and defense-related (C2 – command & control - and intelligence) organizations. Here the dynamic identification and understanding of information sources and providing for interoperability between market participants (players) in a semantic manner are needed;
- War avoidance operations such as peace-keeping, peace-enforcing, non-combatant evacuation or disaster relief operations. In classical war operations the technology of control is strictly hierarchical, unlike operations other than war are very likely to be based on cooperation of a number of different, quasi-volunteered, vaguely organized groups of people, non-government organizations, institutions providing humanitarian aid but also army troops and official governmental initiatives. Here many participants will be ready to share information with some well specified community.

The main project objectives were formulated as follows:

1. *Specification of advanced requirements* for fusion-based knowledge management systems in the scalable infosphere which provide interactive support for the management of large, complex operational environments, such as anti-terror operations, global supply networks, and etc.
2. *The rapid knowledge fusion methodology* into the infosphere.
3. *Integrated framework* of fusion-based knowledge logistics into scalable infosphere.
4. *The multiagent architecture* for fusion-based knowledge logistics technology.
5. *The research software* prototype that should make to validate the major solutions concerning architecture, models and methods of the developed technology.
6. *Case-based evaluation of the developed integrated approach* (methodology, integrated framework, and a set of models and methods) implemented within the prototype.

According to the objectives an approach to knowledge logistics through knowledge fusion has been developed. It is oriented on a configuration of Knowledge Source Network (KSNet-approach) incorporating end-users, tasks, and loosely coupled knowledge sources (experts, knowledge bases, repositories, etc.) distributed throughout the global information environment (infosphere). The approach is based on such advanced technologies as constraints satisfaction / propagation, ontology management, repository parallel development, and intelligent agents.

As results of the developed approach the main outcomes are stated as follows: ontology-driven methodology underline by the model of object-oriented constraint networks for knowledge fusion has been proposed; multiagent architecture of the system implementing the approach has been developed; applicability of the approach has been verified via experimenting with a developed prototype. The experiments were based on a Binni coalition operation scenario for health service logistics. The possible application domains of the approach are domains dealing with distributed (cooperative) problem solving as logistics, supply chain management, configuration management, e-business, e-government, e-health, e-science, etc.

Knowledge logistics tightly correlates with the ideas of Semantic Grid and can be considered as an intelligent service for Knowledge Grid environment (a set of well-organized knowledge together with a set of knowledge management operations). To provide flexible integration with a Grid infrastructure service-based architecture was developed, implemented and tested for the knowledge logistics system. Also the developed approach is closely related to the systems addressing Levels 2+ (Situation Awareness) and 3 (Impact assessment) of the JDL (“Joint Director of Laboratories”) information fusion model oriented to on-the-fly impact assessment of coalition operation including such tasks as coalition configuration, coalition resource management, operation scenarios creation and analysis, situation prediction, etc.

The current report sums up the obtained results. It is organized in the following way. Section 2 introduces the main principles the approach is based on and the system “KSNet” implementing it. Section 3 describes the multiagent architecture of the system. Section 4 is devoted to the depiction of task-oriented agents and used technologies. Evaluation of the approach via experiments with a prototype of the system “KSNet” based on a case study for e-health / e-government operations is given in the section 5. Main results and future work are discussed in the conclusion.

2. FUSION-BASED KNOWLEDGE LOGISTICS: CONCEPT AND METHODOLOGY

This section presents main aspects and ideas considered during the development of an approach to the knowledge logistics.

2.1. KSNet-Approach for Knowledge Logistics

Rapid development of the Internet has allowed users to be exposed to a large amount of information about different problem areas. This enables shifting the quality of intelligent support of the multidisciplinary problem solving on a new level assuming knowledge integration from distributed sources (knowledge bases, databases, documents, knowledge based tools, experts).

In order to this happen, an intensive cooperation and an open real-time knowledge exchange between participants in the global information environment are required, so that the right knowledge from distributed sources can be integrated and transferred to the right person within the right context at the right time for the right purpose. The aggregate of these interrelated activities is referred to as Knowledge Logistics (KL) (Smirnov, 2001b, Smirnov et al., 2003b). KL is based on individual user requirements, available knowledge sources (KSs), and current situation analysis in the information environment. Hence, intelligent support systems operating in this area must react dynamically to unforeseen changes and unexpected user needs, keep up-to-date resource value assessment data, support rapid conducting of complex operations, and deliver results to the users/knowledge customers in a personalized way. An efficient approach to building intelligent support systems is required, which could make possible for the data to become information that leads to knowledge and to sophisticated understanding (Figure 1).

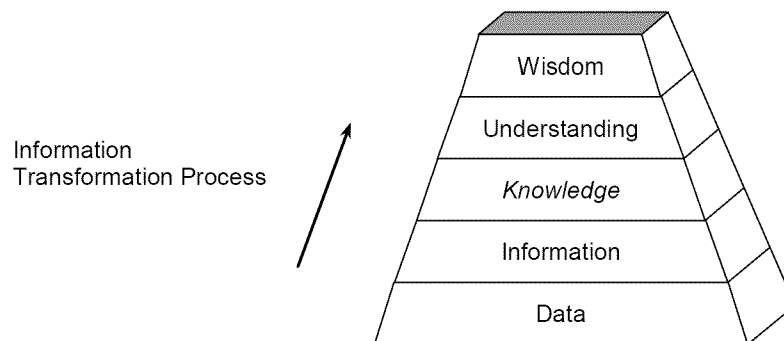


Figure 1. Conceptual framework of information support (adapted from (Anken, 2002))

The current trends require using a global information environment (called “*infosphere*”) including (Smirnov, 2000):

- end-users/customers;
- loosely coupled knowledge sources/resources;
- set of tools and methods for information processing.

Infosphere is called “*scalable infosphere*” when information environment can be scalable against both the number of units and the number of various types of units (Smirnov, 2001a).

Technology of knowledge fusion (KF) is based on the synergistic use of knowledge from multiple distributed sources. The technology can be considered as a basis for KL activities. KF implies synergistic use of knowledge from different resources in order to complement insufficient knowledge and obtain new knowledge.

Network of loosely coupled sources located in infosphere is referred to as “Knowledge Source Network” (“KSNet”). The term KSNet has its origin in the concept of virtual organization based on the synergistic use of knowledge from multiple sources. A KSNet is defined as a flexible connection of appropriate KSs at different locations with the target to fulfill a concrete task. The KSNet exists for a predefined period of time. The network becomes real when a concrete realization takes place or at least the necessary budget is endorsed. During the planning phase until the offer is ratified the KSNet units represent a planning task in order to design and evaluate potential scenario solutions for the decision making of task.

Figure 2 explains roughly basic concepts of KSNet-approach and multi-level KSNet configuration. The upper level represents a customer-oriented knowledge model based on a fusion of knowledge acquired from network units (KSs) that constitute the lower level and contain their own knowledge models.

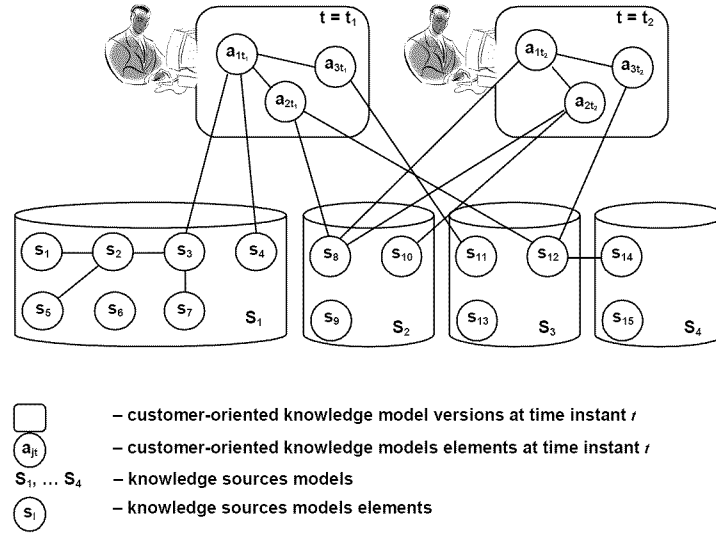


Figure 2. KSNet-approach: distributed multi-level knowledge fusion management as the KSNet configuration

Knowledge management is defined as a complex set of relations between people, processes and technology bound together with the cultural norms, like mentoring and knowledge sharing. Knowledge management consists of the following tasks: knowledge discovery (knowledge entry, capture tacit knowledge, etc.), knowledge engineering (knowledge base (KB) development, knowledge sharing and reuse, knowledge exchange, etc.), knowledge mapping (identifying KSs, indexing knowledge, making knowledge accessible, etc.). The areas of knowledge management, KF system "KSNet" deals with, are presented in Figure 3.

KL as a new direction of knowledge management assumes presence of fusion processes. In (Holsapple and Singh, 2001) the most complete sequence of main operations for knowledge integration, referred to as knowledge chain, is suggested. This offers a base for forming KF operations depicted in Figure 4 and listed below. (It is necessary to emphasize the importance of KF management operation, which is an enabler for the entire fusion process):

- *Capturing knowledge* from KSs and their translation into a form, suitable for supplementary use;
- *Acquisition of knowledge* from external sources;
- *Selection of knowledge* from internal sources (local KBs);
- *Knowledge generation*: producing knowledge by discovering or deriving from existing knowledge;
- *Internalization*: changing system knowledge by saving acquired, selected and generated knowledge;
- *Externalization*: embedding knowledge into system's output for release into the environment;

- *Knowledge Fusion Management*: planning, coordination, collaboration, and control of operations constituting the process of KF.

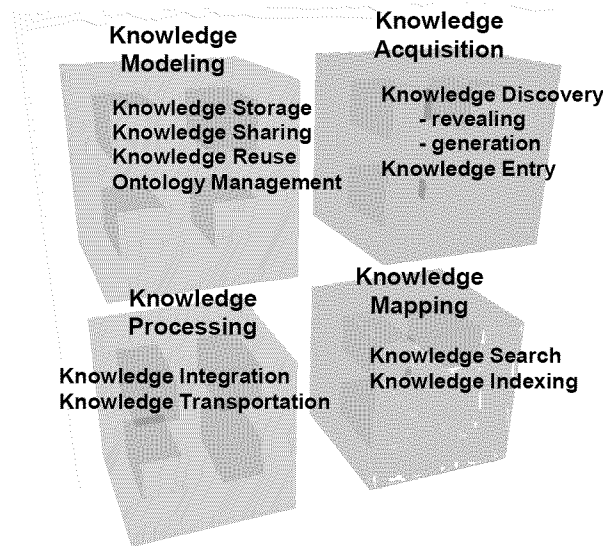


Figure 3. Knowledge management areas of the system "KSNet"

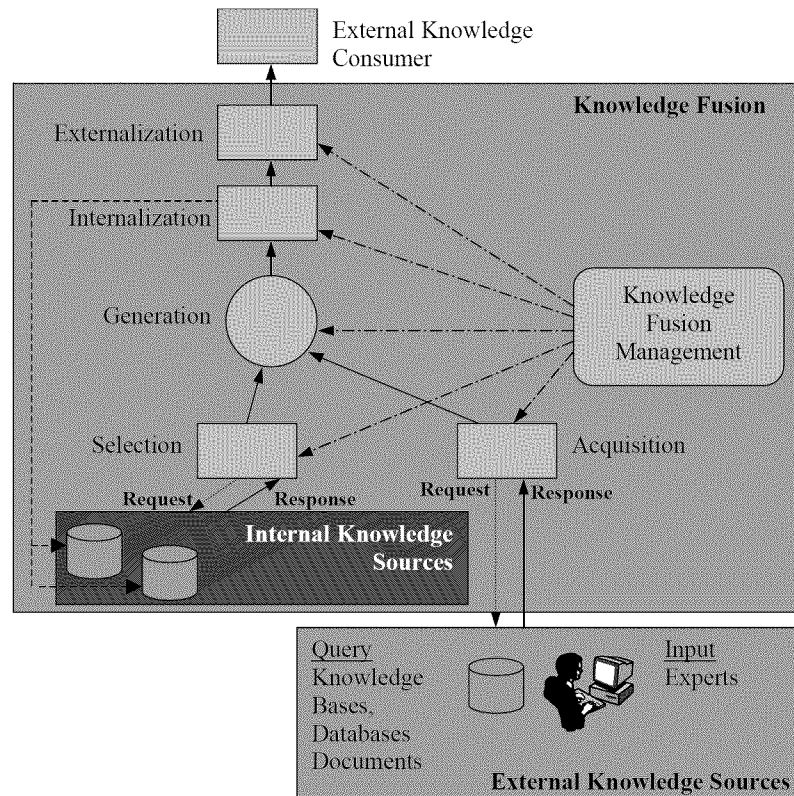


Figure 4. The process of knowledge fusion

In order to improve rapid fusion of knowledge from loosely coupled sources located in infosphere, it is necessary not only to find required sources but also to identify their usefulness for solving real problems. During this it is reasonable to:

- Apply user profiles;
- Offer tips and hints to the user, recognize tacit user interests;
- Utilize techniques of knowledge/ontology reuse;
- Perform indexation of stored knowledge;
- Define importance of the information;
- Increase intelligibility of knowledge representation for the users, involved into the processes of development, edition, updating, etc.

2.2. Modern Requirements and Standards to Fusion-Based Knowledge Management Systems in the Scalable Infosphere

Since KL closely integrates with different knowledge management areas it is to satisfy to modern requirements and rely on common standards used in this area.

2.2.1. Modern Requirements to the Fusion-Based Knowledge Management Systems

KSs include (Figure 5):

- databases and KBs;
- documents stored in Internet/Intranet/Extranet;
- expert knowledge;
- libraries of solved problems case description;
- libraries of standard/typical situation description.

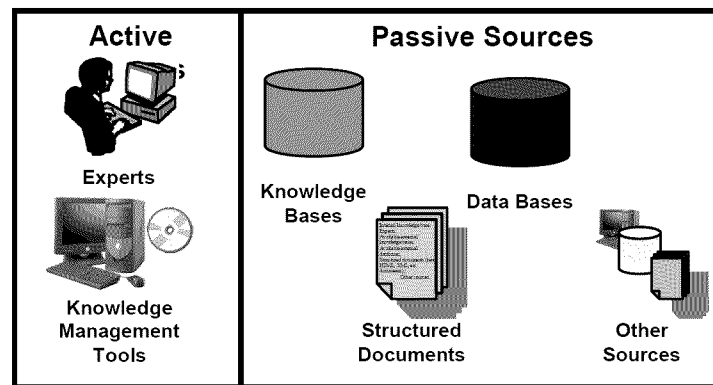


Figure 5. Knowledge sources

Modern requirements to the KF-based systems include:

- *Flexibility.* The system must be ready for sudden changes in the target problem requirements. It maintains its flexibility by keeping minimal information volume in the sources.
- *Learning from the user.* If the user declines a suggested solution or believes that it is not optimal, it is necessary to provide for an ability to include required changes into the system behavior rules.
- *Integrity.* During development of the system it is necessary to perform monitoring of infosphere KSs for their availability and changes. If KS becomes unavailable it is necessary to remove all the references to it

and check knowledge, synthesized while using this source. When the source content changes it is also necessary to perform checks for knowledge consistency.

- *Velocity*. The system permanently seeks for the ways to reduce and/or compensate the variability in customer/user demand and suppliers/sources.
- *Open Connectivity*. Ontologies and KBs built during the process of system utilizing must be available for shared access by external users. Besides, database and KB developers can represent the sources in the required form to expand the set of available KSs.
- *Reasoning*. The system must have clear plan of actions to achieve its goals and reasoning for proposed solutions.
- *Customizability*. The system must be ready to build any possible configuration of knowledge domain model that a customer (user) requests. Besides, it must be able to motivate the suggested solution.
- *Hard' real-time*. The system must have features that guarantee a response within a fixed amount of real-time.

End-user requirements to the knowledge management systems include:

- *User Friendly Interface*. Not all users have the same experience of working with computers. Therefore, it is necessary to have several options of user interface adjustment (from basic to advanced levels) for computer implementation of the system. The basic interface is oriented to the user who poorly knows what he (she) can get from the system. The advanced interface is oriented to the user who feels at home with the system operation details and knows how to build a request in the most efficient way.
- *User Authentication*. The system must contain features of user identification, setting and checking user rights.
- *Access Control*. The system must guarantee the access control in order to prevent unauthorized access to commercial or confidential information.

Applications developed for a work in the scalable infosphere and based on utilizing information from various sources, usually, operate in real-time environment. Given below is the list of requirements to the similar system:

- Clear, real-time graphic displays;
- Web-enabling these displays for real-time update;
- Secure encryption of the Web enabled displays to keep the information out of the wrong hands;
- Ability to collect data, in real time from diverse and dispersed sources such as business systems, industrial computers, sensors, vision systems, and other data sources simultaneously;
- Ability to fuse all these data into meaningful, interconnected, information objects and then fuse this information to knowledge via *intelligent agents*;
- Application of real-time artificial intelligence.

2.2.2. Open Internet Standards

The technologies and standards is a key to the scalable infosphere paradigm. The open Internet standards (Figure 6) that emerged in the last decade are based on protocols that initially had been developed under the client-server network model. As a result of this and due to the rapid acceptance of object-oriented (OO) technologies, different OO frameworks emerged. These offered a high degree of distribution and scalability but relied on proprietary protocols.

Example of extensible and decentralized framework that can work over multiple computer network protocol stacks is SOAP (Simple Object Access Protocol). SOAP is a light-weight protocol for exchanging messages between computer software, typically in the form of software components. SOAP is one of the enabling protocols for Web services.

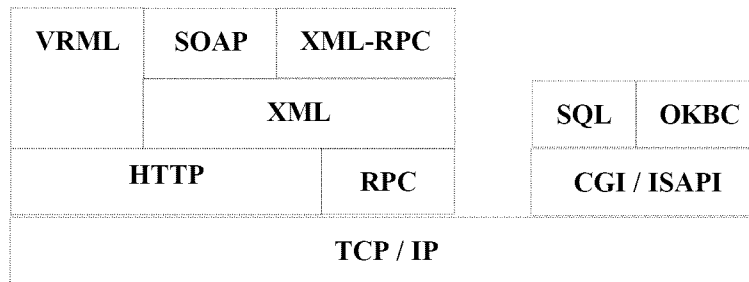


Figure 6. Open Internet standards

XML remote procedure calling (XML-RPC) uses HTTP as the transport and XML as the encoding. XML-RPC is designed to be as simple as possible, while allowing complex data structures to be transmitted, processed and returned. It's a specification and a set of implementations that allow software running on disparate operating systems, running in different environments to make procedure calls over the Internet (XML-RPC, 2003).

- *XML* – Extensible Markup Language, a specification developed by the W3C. XML is a pared-down version of SGML, designed specially for Web documents. It allows designers to create their own customized tags, enabling the definition, transmission, validation, and interpretation of data between applications and between organizations. XML plays a significant role in specifying open Internet value-added protocols for e-commerce applications and for virtual enterprises (Webopedia, 2001).
- *RPC* – A protocol which allows a program running on one host to cause code to be executed on another host without the programmer needing to explicitly code for this. RPC is an easy and popular paradigm for implementing the client-server model of distributed computing. An RPC is initiated by the caller (client) sending request message to a remote system (the server) to execute a certain procedure using arguments supplied. A result message is returned to the caller. There are many variations and subtleties in various implementations, resulting in a variety of different (incompatible) RPC protocols (FOLDOC, 2003);
- *XML-RPC* – A remote procedure call protocol encoded in XML. It is a very simple protocol, defining only a handful of data types and commands. It was first created by Userland Software in 1995, and later picked up by Microsoft. However Microsoft considered it too simple, and started adding functionality. After several rounds of this the standard was no longer so simple, and has re-emerged as SOAP (Wikipedia, 2003).
- *SOAP* – Simple Object Access Protocol. A lightweight and simple XML-based protocol designed to exchange structured and typed information on the Web. The purpose of SOAP is to enable rich and automated Web services based on a shared and open Web infrastructure. SOAP can be used in combination with a variety of existing Internet protocols and formats including HTTP, SMTP, and MIME and can support a wide range of applications from messaging systems to RPC (Remote Procedure Call) (SOAP, 2001);
- *VRML* — Virtual Reality Modeling Language. A specification for the design and implementation of a platform-independent language for virtual reality scene description (FOLDOC, 2003);
- *TCP/IP* — Transmission Control Protocol/Internet Protocol are the main protocols in TCP/IP networks. IP specifies the format of packages, also called datagrams, and the addressing scheme. It allows to address a package and to drop it in the system, but there's no direct link between the sender and the recipient. TCP enables two hosts to establish a connection and exchange streams of data. TCP guarantees delivery of data and also guarantees that packages will be delivered in the same order in which they were sent (Webopedia, 2001);
- *HTTP* – HyperText Transfer Protocol, the underlying protocol used by the World Wide Web. HTTP defines how to form and transmit messages, and what actions Web servers and browsers should take in response to various commands (Webopedia, 2001);
- *CGI* — Common Gateway Interface. A standard for running external programs from a World-Wide Web HTTP server. CGI specifies how to pass arguments to the executing program as part of the HTTP request. It

also defines a set of environment variables. Commonly, the program will generate some HTML which will be passed back to the browser but it can also request URL redirection (FOLDOC, 2003);

- *ISAPI* — Microsoft's programming interface between applications and their Internet Server. Active Servers created with ISAPI extensions can be complete in-process applications themselves, or can "connect" to other services. ISAPI is used for the same sort of functions as CGI but uses Microsoft Windows dynamic link libraries (DLL) for greater efficiency (FOLDOC, 2003);
- *SQL* — An industry-standard language for creating, updating and, querying relational database management systems (FOLDOC, 2003);
- *OKBC* — Open Knowledge Base Connectivity. A protocol for accessing KBs (Chaudhri et al., 1998).

2.2.3. *Web & Grid Services*

The idea of open services arises from the concept of virtual organization. It can be said that "services are oriented to virtualization of resources" (The Globus Project Tutorial, 2003). Services address discovery & invocation of persistent services. Internal implementation of services can be of any nature but their interfaces have to be standardized. In other words services must have:

- standard interface definition mechanisms: multiple protocol bindings, multiple implementations, local/remote transparency for:
 - global naming & references,
 - registration & discovery,
 - authorization,
 - notification;
- multiple hosting targets: J2EE, .NET, "C", etc. supporting:
 - lifetime management,
 - concurrency,
 - manageability.

In Grids, services must also support transient service instances, both created and destroyed dynamically. This is done by introducing a so-called Factory service requested to create a new Grid service instance. Create operation can be extended to accept Grid service specific creation parameters and returns a Grid Service Handle (GSH) providing:

- a globally unique URL,
- uniquely identifies the instance for all time,
- based on name of a home mapper service.

A GSH is a stable name for a Grid service, but does not allow clients to actually communicate with the Grid service. A Grid Service Reference (GSR) is a WSDL document that describes how to communicate with the Grid service:

- contains protocol binding, network address, etc.;
- may expire (i.e. GSR information may change).

The Mapper interface allows a client to map from a GSH to a GSR.

The main conceptual differences between Grid service and Web service are the following:

- Grid services are transient: they maybe created and destroyed on demand, using a Factory. Web services are assumed to be available most of the time.

- Grid services have state and Web-services do not.
- There may be many instances of a grid services at one time – usually only a few of a web service.

The technology of open services is yet on a very early stage of its development. However the performed analysis of publications shows that there has been created a set of core standards and protocols. They are mostly the same for both Web and Grid services. Current increasingly popular standards-based framework for accessing network applications is suggested by such companies as Microsoft (Microsoft, 2003), IBM (IBM, 2003), Sun (Sun, 2003) and others and supported by W3C consortium (W3C, 2003). This framework is used in the system "KSNet". It includes (Figure 7):

- TCP/IP: Transmission Control Protocol/Internet Protocol – the suite of communications protocols used to connect hosts on the Internet;
- HTTP: HyperText Transfer Protocol – the underlying protocol used by the Internet;
- XML: Extensible Markup Language – a specification developed by the W3C for a pared-down version of SGML, designed especially for Web documents;
- SOAP: Simple Object Access Protocol – XML-based RPC protocol;
- WSDL: Web Services Description Language – Interface Definition Language for open services;
- DAML-S: supplies Web service providers with a core set of markup language constructs for describing the properties and capabilities of their Web services in unambiguous, computer-interpretable form. DAML-S markup of Web services will facilitate the automation of Web service tasks including automated Web service discovery, execution, interoperation, composition and execution monitoring.
- UDDI: Universal Description, Discovery, & Integration – a “meta service” for locating open services by enabling robust queries against rich metadata (UDDI.org, 2003).

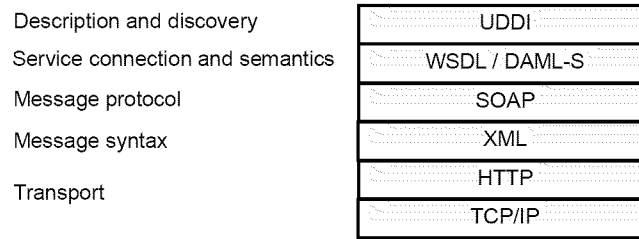


Figure 7. Standards and protocols for open service platform

2.3. Ontology-Driven Methodology of Fusion-Based Knowledge Logistics

2.3.1. Scheme of Knowledge Fusion Support

Since KL is oriented on an identification of user needs and a provision of the user with an appropriate (up-to-date, relevant, and accessible) knowledge, the methodology considers the problem of KL by an example of user request processing. User needs are introduced by a user request. The request formulates a problem to be solved. The requested knowledge (data, information) to a solution can be arrived at is provided by a set of KSs.

A problem solving can demands knowledge of various application domains. Fusion of the knowledge of the domains results in a description of modes (methods) of the current problem solving (knowledge how to solve the problem). Fusion of the knowledge of KSs gives instantiated knowledge meeting the requirements of the solution arriving.

The approach is based on the idea that knowledge can be represented by two levels. The upper level describes the structure of knowledge. It is made up of knowledge describing how to solve a problem. Knowledge represented by the lower level is an instantiation of the upper level knowledge. This knowledge describes a

problem situation or holds object instances. As it is mentioned above knowledge can be fused at both of the two levels (Figure 8).

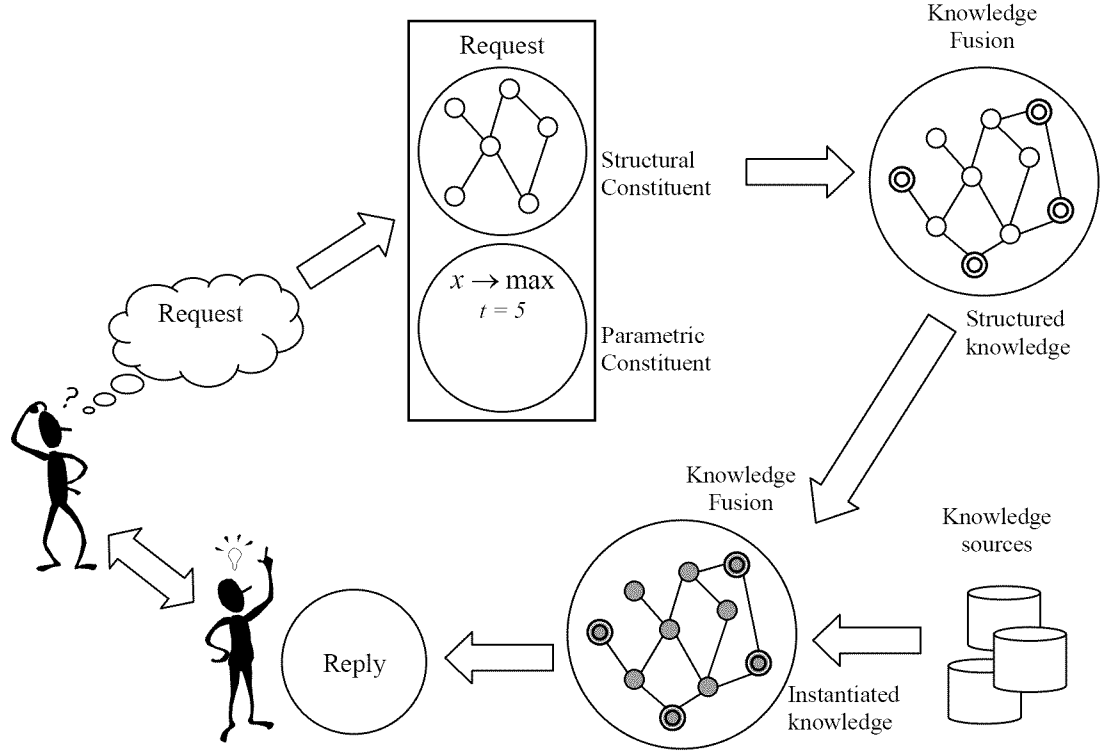


Figure 8. Knowledge fusion support

A user request and KSs can contain knowledge of the both levels. A part of the user request relating to the problem solving knowledge is represented by a structural request constituent. This constituent determines what kind of domain knowledge is to be used in the problem solving. The part of the user request containing problem situations is represented by a parametric request constituent. Knowledge providing modes of the problem solving is formed based on the knowledge represented by the structural request constituent through fusion of the knowledge of various application domains. The parametric request constituent determines requirements that a solution is to meet. Thereby, the problem solving involves fused domain knowledge, instantiated knowledge fused from KSs, and additional restrictions imposed by the.

The KSNet-approach is based on the idea that the lower level knowledge is represented by restrictions on the upper-level knowledge. That was the main reason to use a constraint-based approach to the problem description. With reference to the user request processing the upper-level knowledge is shared knowledge of the user request and KSs. Knowledge corresponding to individual user requirements (the parametric request constituent) and knowledge peculiar to KSs are represented by restrictions on the shared knowledge.

2.3.2. Object-Oriented Constraint Networks as Knowledge Representation Model

The goal of the upper level is to account for what it is observed. This level lies on a formalism chosen for the representation. Since KL deals with distributed and heterogeneous resources the upper level is represented by ontology model ensuring homogeneous and common understandable knowledge representation. Ontology-based knowledge representation enables knowledge reuse, sharing and exchange.

As the fundamental ontology for knowledge representation the model of object-oriented constraint networks (OOCN) coupled with the OOCN notation has been chosen (Smirnov and Chandra, 2000). OOCN is described by sets of objects, attributes, attribute domains, and constraints $OOCN = (\{Object\}, \{Attribute\}, \{Domain\}, \{Constraints\})$. With reference to the ontology representation an ontology (A) is described as:

$$A = (O, Q, D, C)$$

where O – a set of *object classes* (“classes”); each of the entities in a class is considered as an *instance* of the class;

Q – a set of class attributes (“attributes”);

D – a set of attribute domains (“domains”); and

C – a set of *constraints*.

As a constraint-based tool ILOG tool has been chosen (ILOG, 2003]) A constraint satisfaction model supported by ILOG is presented in Figure 9.

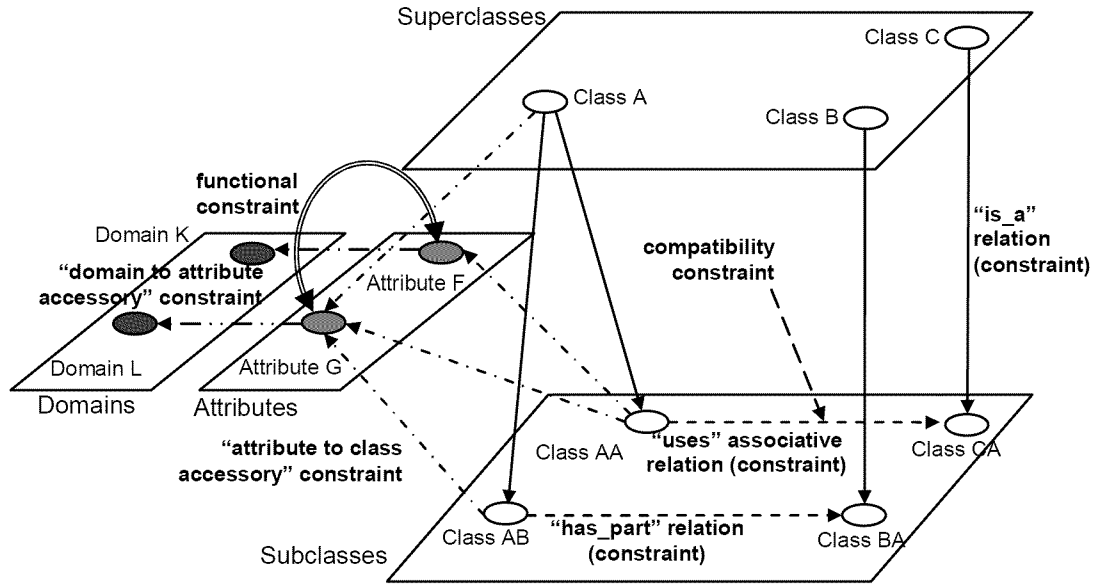


Figure 9. ILOG Constraint Satisfaction Model

Taking into account (1) associations commonly used in the representations of ontology classes relations as hierarchical and associative relationships, (2) constraints inherent in the OOCN model as accessory of attributes to classes and domains to attributes, and functional constraints, and (3) relations supported by the ILOG constraint satisfaction model the set of constraints is defined as:

$C^I = \{c^I\}$, $c^I = (o, q)$, $o \in O$, $q \in Q$ – accessory of attributes to classes;

$C^{II} = \{c^{II}\}$, $c^{II} = (o, q, d)$, $o \in O$, $q \in Q$, $d \in D$ – accessory of domains to attributes;

$C^{III} = \{c^{III}\}$, $c^{III} = (\{o\}, True \vee False)$, $|\{o\}| \geq 2$, $o \in O$ – classes compatibility (compatibility structural constraints);

$C^{IV} = \{c^{IV}\}$, $c^{IV} = \langle o', o'', type \rangle$, $o' \in O$, $o'' \in O$, $o' \neq o''$ – hierarchical relationships (hierarchical structural constraints) “is a” defining class taxonomy ($type=0$), and “has part”/“part of” defining class hierarchy ($type=1$);

$C^V = \{c^V\}$, $c^V = (\{o\})$, $|\{o\}| \geq 2$, $o \in O$ – associative relationships (“one-level” structural constraints);
 $C^{VI} = \{c^{VI}\}$.
 $C^{VI} = f(\{o\}, \{o, q\}) = \text{True} \vee \text{False}$, $|\{o\}| \geq 0$, $|\{q\}| \geq 0$, $o \in O$, $q \in Q$ – functional constraints referring to the names of classes and attributes.

Classes, attributes, domains, and constraints are referred to as the ontology elements henceforth.

The described model is compliance with both frame-based (KIF) (KIF, 1992) and object oriented (DAML+OIL) (DAML+OIL, 2001) approaches to knowledge representations

2.3.3. Uncertainties in Knowledge Logistics

Since KL systems operate in a stochastic environment uncertainties caused by the following conditions (Giachetti, et. al., 1997) are taken into account:

- *data imprecision* – variable value is not necessarily explicitly known from KSs or users,
- *knowledge imprecision* – due to utilization of human (expert) knowledge including reasoning and skills,
- *relationship imprecision* between elements of KSNet,
- *inconsistency imprecision* from conflicting major objectives of various phases of the system "KSNet" operation.

These uncertainties are reasons to implement the fuzzy constraint satisfaction model for such systems.

In accordance with the formalism of object-oriented constraint networks knowledge is described as $CNet = (O, Q, D, C, I)$ where O , Q , D , C are sets of ontology elements and I is an information content (class instances) of the constraint network. Information content is a set of instances with fixed attribute values (known or not).

Fuzzy constraint network is an extension to the above constraint network and can be described as $(O, Q, D, C_\mu, W, T, I_p)$, where

O , Q , and D are the same as above;

C_μ is a set of constraints, where each constraint contains a function μ of membership to $[0,1]$ associated to weight ω_c representing its weight (importance) or priority; W is a weight scheme, i.e. a function combining satisfaction degree of constraint $\mu(c)$ to ω , for estimation of weighted satisfaction degree of $\mu^\omega(c)$; T is an aggregation function, which performs simple partial regulating on defined values, defining C_μ ; I_p is an information content of the constraint network, which has a nondeterministic or probabilistic nature.

Uncertainties are introduced in the set of constraints in the following way:

- A fuzzy relation between classes and attributes can be defined as:

$$R : O \times Q \rightarrow [0,1];$$

$$R\{o_j, q_k\} = \mu(c), o_j \in O, q_k \in Q, c \in C^I.$$

In other words, class o_j and attribute q_k can be connected at the moment t when $\mu(c) \geq \alpha_t^I$ where α_t^I is the system truth threshold defined by expert.

- A fuzzy relation in the compatibility structural constraints, hierarchical structural constraints and “one-level” structural constraints can be defined as:

$$R : O \times O \rightarrow [0,1],$$

$$R\{o_j, o_k\} = \mu(c), o_j \in O, o_k \in O, c \in C^{III} \cup C^{IV} \cup C^V.$$

In other words, classes o_j and o_k should be connected at the moment t when $\mu(c) \geq \alpha_t^i$ where α_t^i is the system truth threshold defined by expert and $i \in \{3,4,5\}$.

- A fuzzy relation of the accessory of domains to attributes can be defined as:

$$R : O \times Q \times D \rightarrow [0,1],$$

$$R\{o_j, q_k, d_n\} = \mu(c), o_j \in O, q_k \in Q, d_n \in D, c \in C^{II}.$$

In other words, attribute q_k belonging to class o_j can be equal to d_n (or have a value from d_n) at the moment t when $\mu(c) \geq \alpha_t^{II}$ where α_t^{II} is the system truth threshold defined by expert. This relation can be described by trapezoidal membership function (Figure 10).

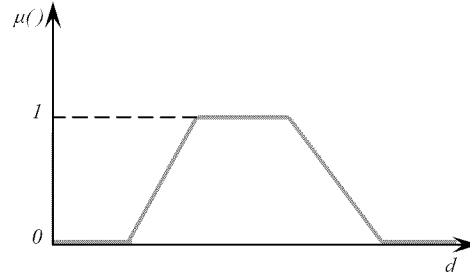


Figure 10. Trapezoidal membership function

- Uncertainties can be penetrated into functional constraints using linguistic variables. In (Zadeh, 1994) it is presented relation between fuzzy and linguistics variables, i.e. “... fuzzy rules and fuzzy graphs bear the same relation to numerically-valued dependencies that linguistics variables bear to numerically-valued variables”. Using verbal scales and linguistic variables an expert can assign a weight to each functional constraint.

Linguistic variables and verbal scales (Wong et al., 2003) could be useful to define constraints contained in user requests. Naturally a user formulates the goal in a verbal style with preferring quantity properties (fast, faster, and fastest) rather than quantity attributes (speed 60-100 miles per hour, 90-180 mph, 150-220 mph). Using the appropriate fuzzy relation (Table 1) it is possible to formalize the goal numerically.

Table 1. An example of verbal and numeric scale to formalize the goal

Verbal goal value	Numeric goal value
Fast	60-100
Faster	90-180
Fastest	150-220

Uncertainties could be useful to extend a list of possible answers to the user and estimate answer relevance. Since any crisp variable is a special case of fuzzy variable it can be generalized that all attribute values of a fuzzy object-oriented constraint network are described by fuzzy variables. E.g., if a user ask for fast vehicle, it is possible to define a restriction to vehicle speed – from 60 miles per hour (an object of class “Vehicle” must have a value of attribute “Speed” more than 60). Using a fuzzy relation of the accessory of domains to attributes it is possible to define that an object of class “Vehicle” having a value of attribute “Speed” between 55 and 60 is also fitting to a user. But quality of this answer is less then 1.

2.3.4. *Ontology-Driven Methodology Principles*

KSNet-approach proposes ontology-driven methodology to knowledge source network configuration (Figure 11). Main components of the KSNet are user and KSs representing various kind of knowledge. The aim of user request processing consists in the configuration of a network of KSs containing information relevant to the user request, generation of an appropriate solution relying on this information, and presenting the solution to the user.

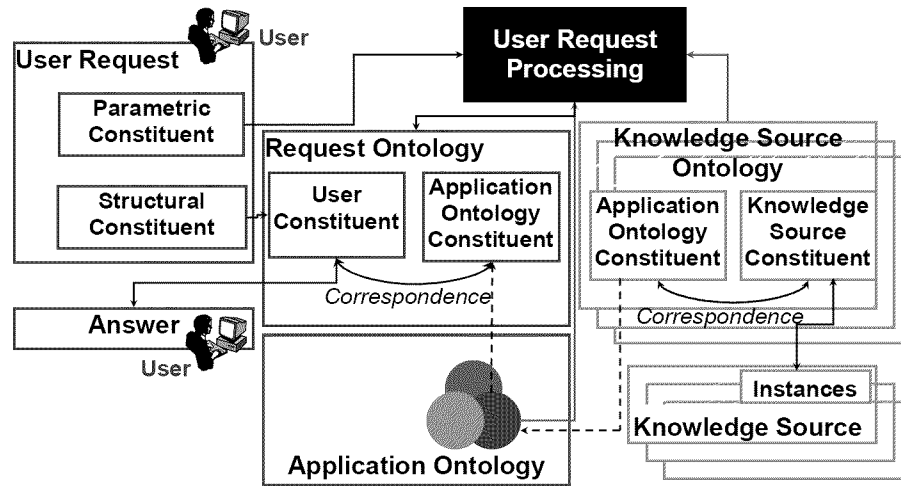


Figure 11. Framework of ontology-driven methodology

The central role in the user request processing is featured to application ontology (AO). It represents the ontology describing a problem formulated in the user request and a way of its solving. AO describes both knowledge related to the user request (user constituent in Figure 11) and knowledge representing requested information containing in KSs (knowledge source constituent in Figure 11). AO is formed as a user request has entered the system. Next, a set of KSs containing the requested information is associated with the same AO. Thereby AO represents a shared knowledge of the user and KSs. The KSNet-approach is based on the idea that knowledge corresponding to individual user requirements and knowledge peculiar to KSs are represented by restrictions on the shared knowledge described by AO (Smirnov et al., 2003c).

The parts of a user request and KS corresponding to ontology elements are represented by the structural constituents of the user request and KS respectively. The parts relating to values and hence to instances are represented by the parametric constituents of the user request and KS. The request processing corresponds to the search for a solution by ILOG. For that the parametric constituents and AO are processed by ILOG as a constraint satisfaction problem. The found solution is transmitted to the user.

At the level of the user request processing knowledge of different KSs, providing knowledge to the request, is used. The result of the user request processing is a new knowledge obtained by fusion of knowledge from

several KSs. The result can be considered as a new KS. Structured knowledge of this new KS is represented by means of OL.

AO that is able to describe user needs (the problem), methods solving them, and the information provided by KSs is formed based on the following principles:

1. AO is a specialization of domain and tasks & methods ontologies [Guarino, 1998].
2. AO, domain, and tasks & methods ontologies are represented by means of the same knowledge representation model, notation, and vocabulary.
3. AO is aware of the user's and KSs' terminologies.
4. The problem represented by AO can be processed as the constraint satisfaction problem.
5. AO is formed on-line as a user request has entered the system.
6. AO is reusable for the solving same and similar problems.

This is implemented through an ontology library (sec. 2.4.1).

2.3.5. Knowledge Fusion Patterns

The following generic KF patterns have been advanced for KF operations (Figure 12):

- *Selective Fusion*. New KS is created which contains required parts of the initial KSs. Initial KSs preserve their internal structures and autonomy.
- *Simple Fusion*. New KS is created which contains initial KSs. Initial KSs preserve their internal structures and lose (partially or completely) their autonomy.
- *Extension*. One of initial KS is extended so that it includes the required part of other initial KS which preserves its internal structure and autonomy.
- *Absorption*. One of initial KSs is extended so that it includes other initial KS which preserves its internal structure and loses (partially or completely) its autonomy.
- *Flat Fusion*. New KS is created which contains initial KSs. The initial KSs dissolve within new KS and do not preserve their internal structures and autonomy.

Developed KF patterns are illustrated via the following example. Two initial KSs (A and B) with some structures of primary knowledge units are given. There is a tacit relationship between two primary knowledge units, namely a3 from A and b2 from B. It is necessary to fuse two sources preserving the internal knowledge structure and revealing the above tacit relationship.

2.4. Knowledge Repository Structure

Due to large amount of knowledge involved in the user request processing this knowledge is organized into storages. Besides *ontology library* the system structure includes *internal knowledge base*, *knowledge map*, and *user profiles* as components of the knowledge repository. The repository is organized composed of the following structural components (Figure 13): (i) a semantic component used for knowledge representation in a common notation and vocabulary; (ii) a service component used for knowledge indexing and search; (iii) a physical component used for knowledge storage and reuse.

The *semantic component* is represented by the ontology library (OL). The *service component* is represented by the knowledge map and user profiles. The *physical component* is represented by the internal knowledge base.

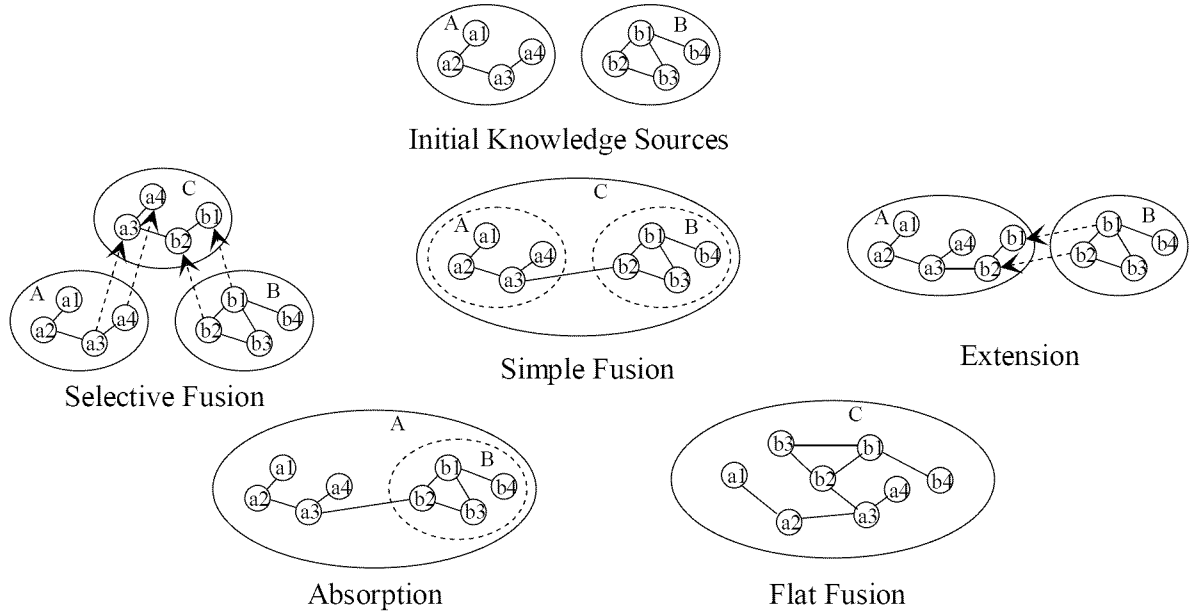


Figure 12. Knowledge fusion patterns

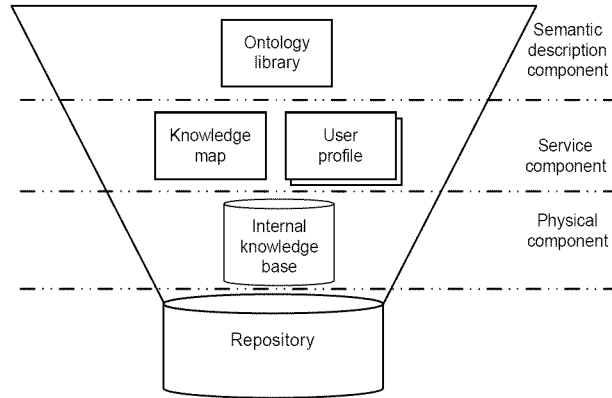


Figure 13. Knowledge repository structure

2.4.1. *Ontology Library*

OL is a central knowledge storage assigning a common notation and providing a common vocabulary to ontologies it stores. As the common notation the formalism of OOCN is used. As the common vocabulary the lexical reference system WordNet [WordNet, 2003] is chosen. The usage of OL enables to share and reuse ontology, and versioning them.

Main components of the OL are domain, tasks & methods, and application ontologies. All these ontologies are interrelated according to N. Guarino proposal [Guarino, 1998] in such a way that AO is a specialization of both domain and tasks & methods ontologies. As the top-level ontology the model of OOCN is used.

According to the chosen formalism the tasks and methods of the tasks & methods ontology are described by ontology classes, output and input parameters of the tasks/methods are described by attributes of these classes. Alternative methods are represented by "is-a" relationship. The sequence of methods in a task solving is

represented by “part-of” relationship between methods. If an attribute value from a domain ontology can be considered as a parameter of a task/method then an associative relationship is established between a domain ontology class holding the attribute and a class of the tasks & methods ontology representing the task/method and vice versa.

AO is formed as a user request has entered the system. It is based on knowledge of domains and application domains described by an assembly of domain and tasks & methods ontologies. The procedure of forming AO is composed of operations over ontologies as *creation*, *finding*, *slicing*, *merging*, *modifying*, *pruning*, and *validation* specified in (Interim Report # 2, 2001). These operations are possible due to the common representative aids. The procedure consists in an identification of slices of domain and tasks & methods ontologies containing knowledge relevant to the user request with subsequent merging the slices into AO. Since ontologies of different domains are stored in the OL (every domain is represented by its domain ontology) AO can specialize knowledge of several domains. Therefore it represents fused domain knowledge and serves as a cross-domain ontology.

Domain ontologies and tasks & methods ontologies are formed as a new knowledge become available. The new knowledge here is knowledge provided by experts, retrieved from KSs, or obtained as results of user request processing. Both new ontologies can be created (if there is no ontology relating to domain/task/method of the new knowledge) and existing ontologies can be expanded (otherwise).

Besides the listed ontologies OL stores preliminary user request ontology (PRO), user request ontology (RO), preliminary knowledge source ontology (PKSO), and knowledge source ontology (KSO).

PRO describes the user request in the user’s terms (which are used by the user for the requests input) and in the notation of OL. RO is the request representation in terms of the vocabulary and notation of OL. PRO is formed based on the results of the parsing the user request (Interim Report # 3, 2002). The result of the parsing is a set of the request terms separated according to its belonging to the structural and parametric request constituents. In PRO creation only the structural constituent takes part. RO is formed by the alignment of PRO and AO.

KSs providing knowledge for the user request are represented by PKSOs and KSOs. PKSO describes the knowledge of KS in terms of this KS and in the notation of OL. KSO is the representation of KS in terms of the vocabulary and notation of OL. The same way as the forming RO KSO is formed by the alignment of PKSO and AO. Relationships between the ontologies of OL are presented in Figure 14.

Each request is represented by its PRO (the relationship one-to-one in Figure 14 between request and PRO). Since RO creation corresponds to the alignment of AO and PRO, each PRO is related to one AO and one RO. In its part, single AO can cover several ROs and correspondingly several PROs (one-to-many relationships between AO and RO, and AO and PRO in the figure). If there is in OL AO covering RO for the request entered the system this AO is reused. The associative relationship between RO and tasks & methods ontology illustrates the case when methods resolving semantic conflicts (e.g., “class—attribute—attribute value” mismatch, conflicts in classes subordination [Chaudhri et al., 2000], etc.) between AO and PRO are attached.

The one-to-many relationship between AO and KSO is explained by the fact that several KSs can be found for a request. The same way as every request has its preliminary RO, every KS is described by its preliminary KSO. The relationship one-to-many in Figure 14 between preliminary KSO and KSO is refers to by the fact that information from KSs relevant to the particular request and correspondingly to the certain AO is used. Preliminary KSO can contain extra knowledge with regard to such AO and request. Relevant to AO information is described by a part of preliminary KSO represented as a self-contained KSO. The associative relationship between tasks & methods ontology and KSO has the same meaning as in the case with RO.

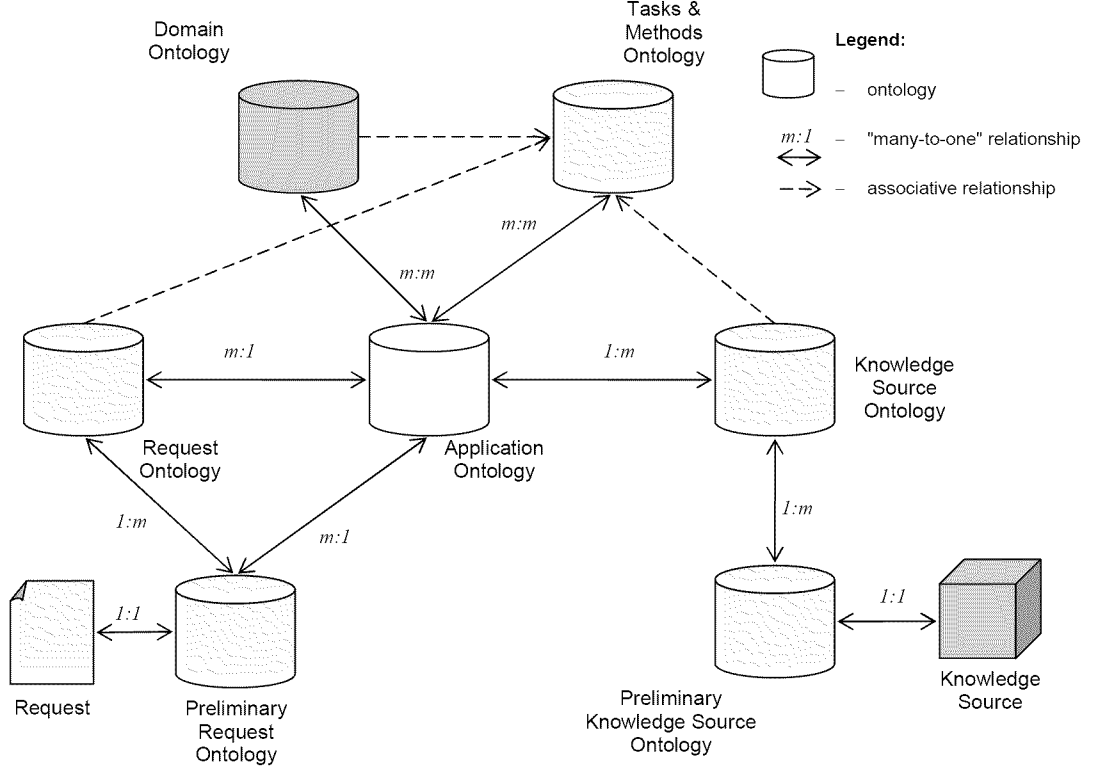


Figure 14. Relationships between ontologies in the ontology library

2.4.2. Knowledge Map

A *knowledge map* as a distributed knowledge storage includes information about locations of KSNet units and information about alternative sources containing similar information and KSs characteristics. Monitoring tools perform permanent checking of KSs availability and perform appropriate changes in the knowledge map. The knowledge map is meant to facilitate and speed up the process of the KSs choice.

Knowledge map is a component of the repository developed for knowledge indexing. It contains (i) KS characteristics, (ii) information about relations between pairs of classes and their attributes from AO and KSs, and (iii) information about alternative KSs.

Requirements and starting conditions for knowledge map creation:

1. AO creation precedes the knowledge map creation. When AO is modified, appropriate changes are made in knowledge map. When AO is archived, an appropriate part of knowledge map is also archived.
2. Creation and modification of knowledge map is done concurrently with creation and modification of KSO.
3. Knowledge map versioning assumes AO version tracking: when a version AO is saved, a corresponding part of knowledge map is also saved; when a version AO is restored, a check and validation of corresponding information in knowledge map is performed.

During KSO creation and modification, correspondence between KS terms and the AO terms is identified. As a result of this process a set of corresponding KSs is defined for classes and their attributes from the AO. In other words, a set of pairs $\{(o_i, q_j) \mid o_i \in O, q_j \in Q, \exists c_n \in C^I : c_n = (o_i, q_j)\}$ from the AO is connected with corresponding KSs (Figure 15).

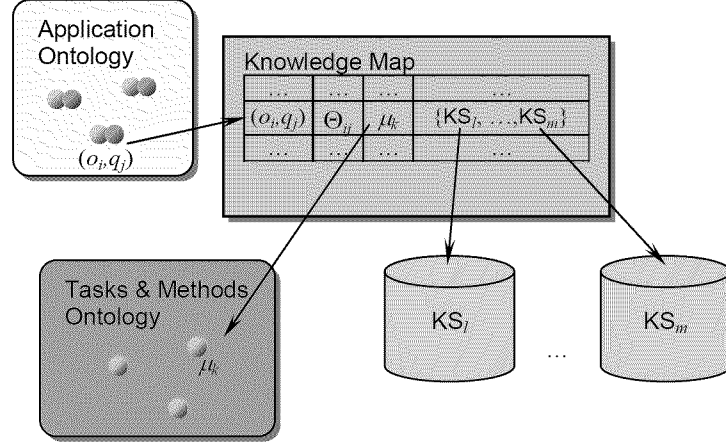


Figure 15. General scheme of knowledge map

The following situations are possible:

1. Two or more alternative KSs correspond to one pair (o_i, q_j) from AO: knowledge can be acquired from any of this KSs meeting user defined constraints included in possible KSNet configuration;
2. Two or more KSs containing different knowledge correspond to one pair (o_i, q_j) from AO: e.g., it is necessary to process knowledge from some of the KSs and to perform predefined operations on received values (conjunction, disjunction, composition, etc.) to find a required attribute value (Desachy, Roux, and Zahzah, 1996, Hunter, 2000, Oussalah, 2000). In this case a method stored in tasks & methods ontology for knowledge integration in accordance with the information stored in knowledge map is chosen and used during fusion of the obtained knowledge.

When the correspondence described by knowledge map is established domains of the attributes values obtained from KSs are to be defined and stored in knowledge map. These domains can be both discrete and interval.

The knowledge map for OL at time instant t is denoted as a set $KMap_t = \{KMapEl_{t_l}\}_{l=1}^{N_{KMap}}$, where $N_{KMap} \in \mathbb{N}$. $KMapEl_{t_l}$ is a tuple: $\langle AOID, (o_i, q_j), \{KSID_p\}_{p=1}^m, \mu_k, \Theta_{ij} \rangle$, where: $AOID$ is the AO identifier; $\mu_k \in TMO$; $m \in \mathbb{N}$ is a number of KSs which have to be processed to obtain the value of the attribute q_j from the pair (o_i, q_j) ; and Θ_{ij} is domain of the attribute values which can be obtained from KS $KSID$. If μ_k is empty, m is set equal 1: $m = 1$. Storage of domains in knowledge map simplifies user request templates creation and allows definition of attributes values domains used in these templates. The time instant t is a discrete value in a sequence t_1, \dots, t_n , where t_1 is a time of the first KSO creation, t_n is time of the recent change of knowledge map. A shift from t_i to t_{i+1} occurs when (i) a new KSO is built or an existing KSO is changed, (ii) KS content or/and KSO are modified.

2.4.3. User Profiles

A *user profile* is an organized storage of information about a user, the user requests history, etc. This element is used for a number of purposes (faster search due to analysing and utilizing request history and user preferences, Just-before-Time request processing, etc.).

The following user types were defined in the system "KSNet": (i) knowledge customers, (ii) experts, (iii) ontology engineers, (iv) knowledge engineers, (v) software engineers and (vi) administrator. The users are divided into 2 classes: (i) users served by the system (knowledge customers), and (ii) users serving the system (all other types of users). The set of users is denoted as $U = \{u_i\}_{i=1}^{N_U}$, where $N_U \in N$ is a number of registered users, and u_i is the user profile described by the tuple: $\langle UserID, PD, SD, KD \rangle$ (Figure 16), where:

1. $UserID$ – a users identifier;
2. PD – personal data: user name, e-mail, home page, demographic data, contact information, etc;
3. SD – system data: login; password; user type; individual attitudes, preferences and settings (including interface language and settings, utilized software and platform, etc.); mail box – storage of system messages (to provide for a "messaging" service).
4. KD – collected data: $KD = \langle H, OtherParameters \rangle$, where:
 - 4.1. H – contact history: sessions protocols (date and time of registration in the system, performed actions, etc.);
 - 4.2. $OtherParameters$ – additional set of parameters associated with the user type:

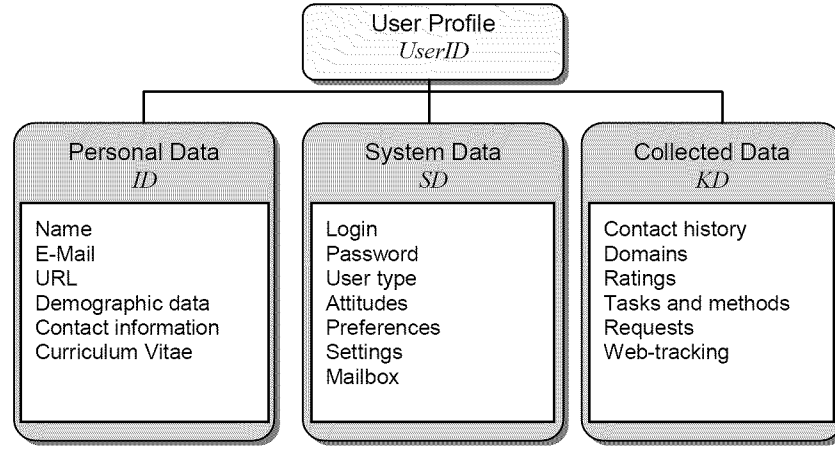


Figure 16. User profile structure

Groups of users with similar interests can be selected by analyzing the users profiles. The set of these groups is denoted as $G = \{g_i\}_{i=1}^{N_G}$, $N_G \in N$, where g_i is a target group. The sets U and G are related as "many-to-many" ($m:m$).

2.4.4. Internal Knowledge Base

Internal knowledge base is used for storage and verification of knowledge (i) entered by experts, (ii) learnt from users (knowledge consumers), (iii) obtained as a result of the KF process, (iv) acquired from KSs which are not free, not easily accessible, etc. (Figure 17).

Users serving the system search for KSs (i) to acquire knowledge for users requests processing and (ii) to provide for a Just-Before-Time¹ function. When a new KS is found (i) its characteristics are saved in knowledge map and (iii) its PKSO is created. Administrator enters the information about the found KSs into the system.

¹ Just-before-Time (JBT) servicing assumes use of internal mechanisms based on anticipation techniques which can make products/services available for customers shortly before they are demanded.

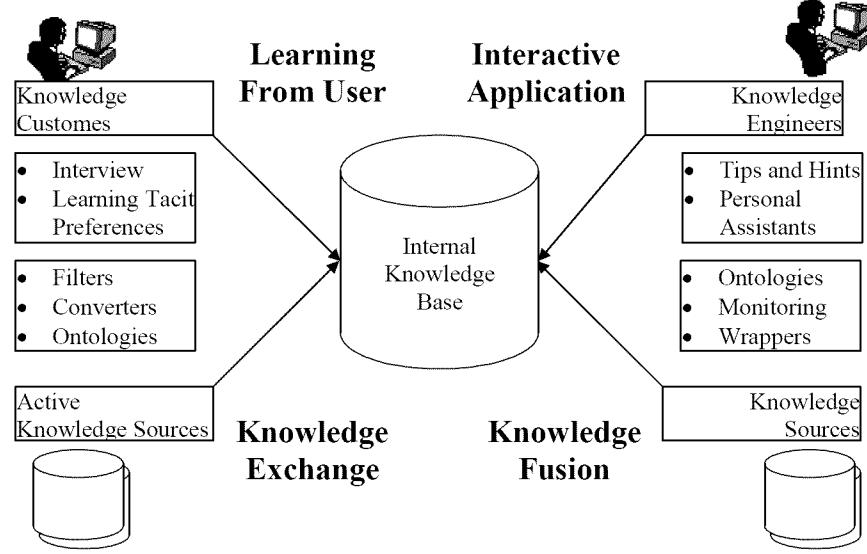


Figure 17. Internal knowledge base

Internal knowledge base contains structured knowledge in the system notation. It has an extensible PKSO describing the structure of stored knowledge, and like other KSs its description in knowledge map. Internal knowledge base is implemented as a set of auxiliary tables, which are a part of the system. Internal knowledge base is used as:

1. Knowledge storage (as KS)
 - 1.1. knowledge stored in chargeable KSs (for user request processing costs reduction);
 - 1.2. knowledge stored in KSs difficult for accessing (for user request processing time reduction and reliability increase);
 - 1.3. knowledge entered by experts;
 - 1.4. new knowledge received as a result of KF;
2. Subrequests with answers storage (for user request processing time reduction and Just-Before-Time service)

Knowledge storage is denoted as $I^{IKB} = \{i_j^{IKB}\}_{j=1}^{N_I}, N_I \in N$, where i_j^{IKB} are instances of classes. For each i_j^{IKB} a list of attributes with associated known / partially known (domains) / unknown values $\{q, v\}_{kj}^{IKB}$ is attached. These sets can be attached to subrequests and used when an appropriate subrequest is activated.

Information about KSs is presented as $\{S_i\}_{i=1}^{N_{KS}}, N_{KS} \in N$, where N_{KS} is a number of attached KSs, and $S = (KSID, KSType, KSParameters, PKSOID, KSOtherParameters)$, where

- $KSID$ – KS's identifier;
- $KSType = IKB \mid Expert \mid ExternalKB \mid DB \mid Document \mid \dots$ – a KS type;
- $KSParameters$ is a set of auxiliary parameters. Its content depends on the KS type:
 - $IKB = \emptyset$,
 - $Expert - UserID$ (expert identifier),
 - $Document$ – Uniform Resource Identifier (URI),
 - DB – database management system (DBMS), location, description, structure,
 - EKB – external KB location, description, language, knowledge representation formalism, development

- environment,
- *PKSOID* – PKSO identifier;
- *KSOtherParameters* - a set of additional parameters, described in the (Interim Report # 1, 2001, Interim Report # 2, 2001):
 - Quantifying:
 - duration of KS accessibility;
 - costs related to knowledge acquisition from commercial sources;
 - completeness: degree of KS translation into the AO terms;
 - rating based on expert group work on ranking alternative sources according to given criteria.
 - Qualifying:
 - availability of required knowledge;
 - newness: date of recent KS update;
 - rating based on previous source utilizing – rating, identified as a result of system's functioning. During the work responses for the same request can be obtained from different sources. This parameter illustrates degree of user satisfaction by the system responses.

3. MULTIAGENT ARCHITECTURE AND RESEARCH PROTOTYPE OF THE SYSTEM “KSNET”

To design the above described KL approach it was necessary to choose effective implementation technology. Agent-based technology is a good basis for KL since agents can operate in a distributed environment, independently from the user and apply ontologies for communication, knowledge representation and exchange. This section describes the tasks solved during the knowledge logistic system “KSNet” prototype development: a design of the agents’ model, choice and modification of the agents’ negotiation protocol, an elaboration of UML-based conceptual projects of agents’ scenarios, and an implementation of the agents community by the multiagent toolkit MAS DK.

3.1. Agents Community of the System “KSNet”

Because of the distributed nature of the problem the KSNet-approach assumes an agent-based architecture. A multiagent system architecture, based on FIPA Reference Model (FIPA, 2002) as an infrastructure for definition of agent properties and functions, was chosen as a technological basis for the system “KSNet” since it provides standards for heterogeneous interacting agents and agent-based systems, and specifies ontologies and negotiation protocols to support interoperability in specific application areas. FIPA-based technological kernel agents used in the system are: wrapper (interaction with KSs), facilitator (“yellow pages” directory service for the agents), mediator (task execution control), and user agent (interaction with users). The following problem-oriented agents specific for KL, and scenarios for their collaboration were developed: translation agent (terms translation between different vocabularies), knowledge fusion (KF) agent (knowledge fusion/integration operation performance), configuration agent (efficient configuring of KSNet), ontology management agent (ontology operations performance), expert assistant agent (interaction with experts), and monitoring agent (KSs verifications).

A major set of agents is represented in Figure 18 in accordance with the above described principles and functions of the KF system.

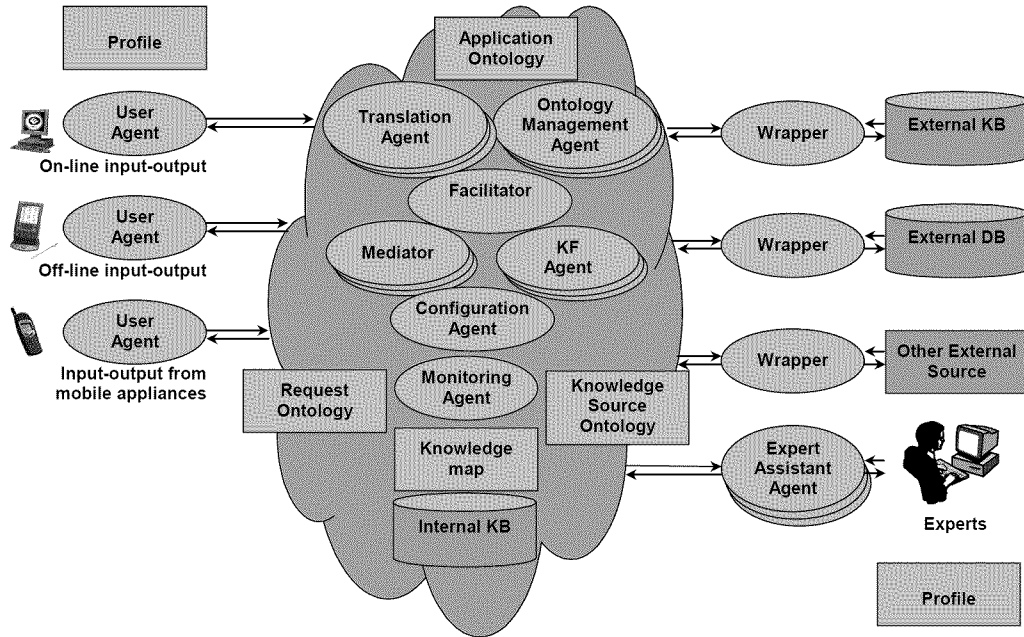


Figure 18. Basic components of multiagent environment of rapid knowledge fusion system

Agents of the system “KSNet” have the following characteristics: (i) benevolence – willingness to help each other, (ii) veracity – agents do not process knowingly false information, and (iii) rationality – willingness to achieve the goal and not to avoid it.

According to the agents’ classification (Brustoloni, 1991) the system agents are divided into three groups (Figure 19):

- *regulation agents*: (i) wrapper, (ii) translation agent;
- *planning agents*: (i) mediator, (ii) facilitator, and (iii) monitoring agent;
- *adaptive agents*: (i) KF agent, (ii) expert assistant agent, (iii) user agent, (iv) configuration agent, and (v) ontology management agent.

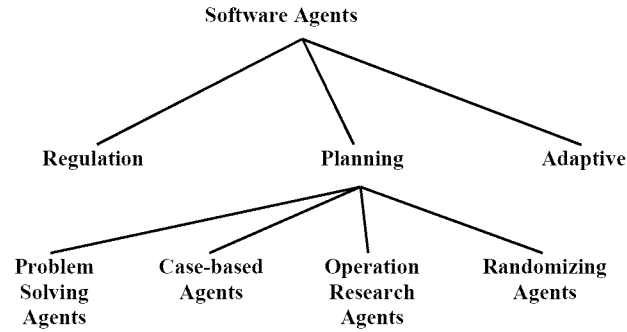


Figure 19. Classification of the software agents

The agents operate during the system “KSNet” life cycle. Expert assistant agents, user agents, facilitator, mediators, translation agent, ontology management agent and monitoring agent create user profiles, support the users' activities and prepare the knowledge representation. The software engineers and experts create software modules and attach new agents to the system. All the types of developed agents are used for (i) user requests processing, (ii) renewing internal knowledge base with active sources (experts, knowledge management tools, and the knowledge engineers), and (iii) diagnosing state of the infosphere.

Depending on problem domain specifics most of existing multiagent systems have their own set of agents with special functions, language and communication rules. Table 2 describes some special features of the proposed agents.

The UML-based state chart diagram of the user request processing scenario is given in Figure 20. This scenario was described in details in (Interim Report # 3, 2002). All stages are supported by the developed agents.

The system's operation starts with entering of a user request. The entered request is recognized and translated from the user notation into the system notation of object-oriented constraint networks. The result of this operation is PRO. After that a new AO, based on AOs, domain ontologies and tasks & methods ontologies stored in OL, or a new AO, coinciding to the PRO in terms of the OL vocabulary, is built.

When AO is created alignment of the elements from PRO and AO is performed in accordance with the rules described in (Interim Report # 2, 2001, section 3). The result of this operation is RO.

After that the operation of KSs search is performed. This assumes analysis of the correlation between KSs content described in knowledge map and classes & attributes of AO. For each found KS, PKSO and KSO are created as a result of AO and PKSO elements alignment. The AO is compared against other existing AOs in OL. In accordance with the results of this analysis, AO can be added in OL.

Table 2. Features of agents of rapid knowledge fusion systems

Agent	Life time	Quantity	General tasks
Wrapper	KS life time	Number of KS types	Translates knowledge from source format representation into system format representation and sends requests from system to source.
Mediator	Task execution time	Number of tasks being processed	Tracks out task processing step-by-step from input to result. Provides negotiations with other agents. Stores temporary results.
Facilitator	System life time	1	Provides a “yellow pages” directory service for the agents.
User Agent	As long as user is registered in the system	Number of registered users	Provides a set of functions for user profile processing. Facilitates request input, provides a set of tips and hints for user, passes messages and information from the system to the user.
Translation Agent	System life time	Number of request input interfaces.	Provides translation functions between user and system. Works with templates and AO.
Expert Assistant Agent	As long as expert is registered in the system	Number of registered experts	Facilitates the process of expert knowledge input into the system. Forms expert profile.
Configuration Agent	System life time	1	Configures KSNet from different sources using knowledge map. Performs scheduling functions. Negotiates with KF agents, wrappers and expert assistant agents.
Knowledge Fusion Agent	System life time	Depends on problem domain	Obtains knowledge from mediator and fuses it. Generates new knowledge. Validates it. Using different KF patterns can perform some operations concurrently.
Monitoring Agent	System life time	1	Provides a set of functions for diagnostics of internal system knowledge repository and external KSs.
Ontology Management Agent	System life time	Depends on problem domain	Provides a set of function for ontology engineering and operation – creation of ontologies for new KSs, modification of AO, etc. Checks correspondence between KS and request ontologies and AO.

The information from KSs is processed by the system using constraint satisfaction techniques in accordance with built KSNet configuration. The latter is built so that the request can be processed in the most efficient way against the user defined criteria (such as time and/or costs) and includes such information as when and which KSs should be used for the user request processing.

The user request processing is finished when the request processing answer is presented to the user. This intends a translation of the AO terms and notations into the PRO terms and notations. The translation is performed based on terms alignment described by RO.

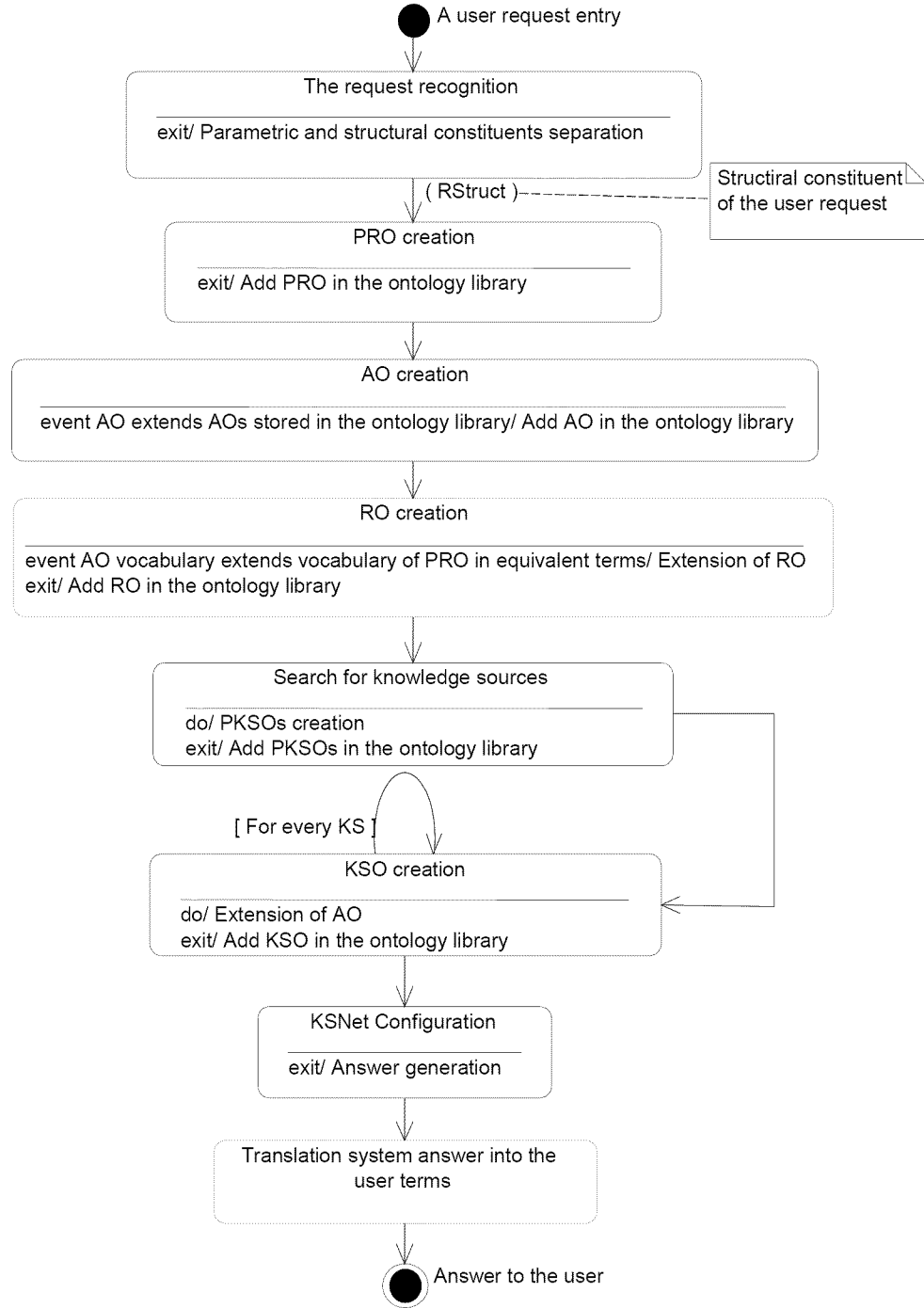


Figure 20. General scenario

In the system “KSNet” the KF takes place during performance of a number of tasks. Use of the KF patterns accelerates the KF process due to typification of fusion schemes.

Table 3 presents KF patterns and system agents used for major KL operations performance.

Table 3. Knowledge fusion patterns and system agents used for major KL operations performance

Operations	KF Pattern	Agents
Ontologies library creation	<ul style="list-style-type: none"> • Simple fusion • Extension 	<ul style="list-style-type: none"> • User agent • Expert assistant agent • Mediator • Monitoring agent • Ontology management agent • Facilitator
Application ontology creation	<ul style="list-style-type: none"> • Selective Fusion • Extension 	<ul style="list-style-type: none"> • Same as above
Request ontology creation	<ul style="list-style-type: none"> • Absorption • Selective Fusion • Extension 	<ul style="list-style-type: none"> • Same as above
KS introduction into the system and KSO creation	<ul style="list-style-type: none"> • Absorption • Selective Fusion • Extension 	<ul style="list-style-type: none"> • Same as above
Knowledge map creation	<ul style="list-style-type: none"> • Extension 	<ul style="list-style-type: none"> • User agent • Expert assistant agent • Mediator • Monitoring agent • Facilitator
New knowledge entry by experts	<ul style="list-style-type: none"> • Extension 	<ul style="list-style-type: none"> • User agent • Expert assistant agent • Mediator • Monitoring agent • Facilitator • Translation agent • Ontology management agent
User request processing	<ul style="list-style-type: none"> • Flat fusion • Extension 	<ul style="list-style-type: none"> • User agent • Mediator • Monitoring agent • Facilitator • Translation agent • Ontology management agent • Wrapper • Configuration agent • KF agent
KF operation – resolution of constraint network defined by partial AO for request processing	<ul style="list-style-type: none"> • Flat fusion 	<ul style="list-style-type: none"> • Mediator • KF agent
Storing of the results of user request processing	<ul style="list-style-type: none"> • Extension 	<ul style="list-style-type: none"> • Mediator • Monitoring agent • Facilitator

3.2. Multiagent Architecture

3.2.1. Agent Structure

Agent structure contains the following modules (Figure 21): (i) identifying, (ii) functional, and (iii) knowledge repository. Identifying module contains such parameters as unique identifier, type, etc. Functional module contains a set of procedures to be executed by an agent this module belongs to. Knowledge repository contains special information, such as history of the agent contacts, temporary results, new knowledge, etc. Agent actions can be described using finite state automata starting when one or more predefined conditions are met (an agent receives a message from another agent or from the system).

Identifying and functional agent modules are shown in Figure 22.

The agents' functions can be divided into the following sets (Table 4):

1. Sensor functions receiving messages from other agents of the system “KSNet” and from other external systems or infosphere units;

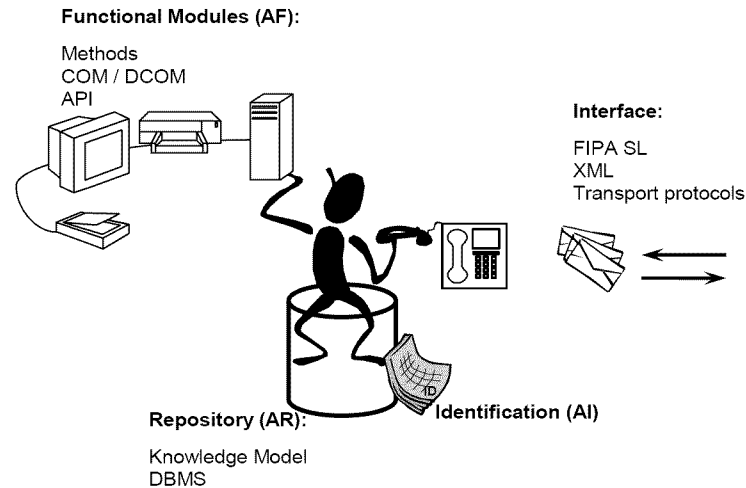


Figure 21. Modular agent architecture

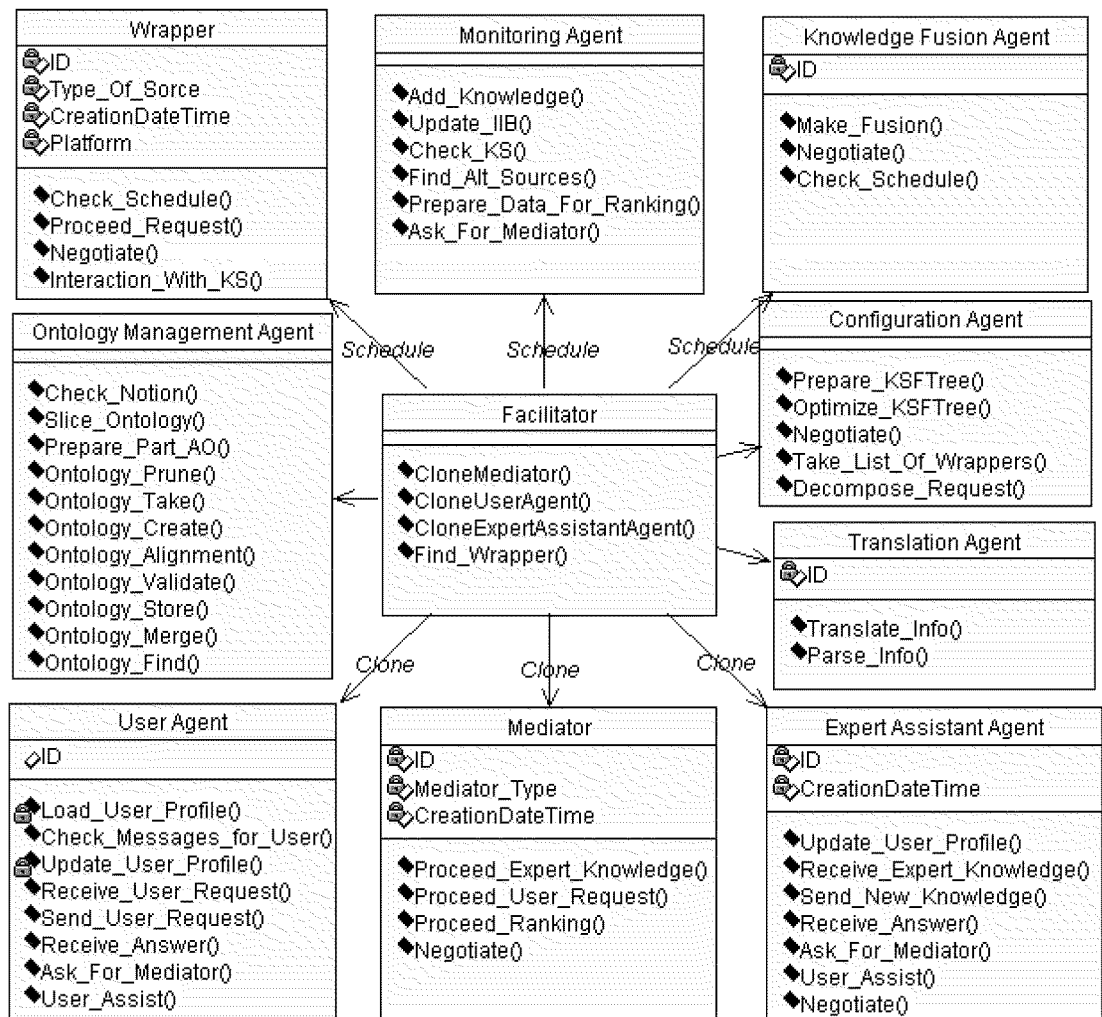


Figure 22. List of the system “KSNet” agents with main properties and functions

Table 4. Agents’ functions in the system “KSNet”

Agents	Sensor functions	Problem-oriented functions	Internal system functions
Wrapper			☑
Mediator			☑
Facilitator	☑		☑
User agent	☑		☑
Translation agent		☑	☑
Expert assistant agent	☑	☑	☑
Configuration agent		☑	☑
Knowledge fusion agent		☑	☑
Monitoring agent	☑	☑	☑
Ontology management agent		☑	☑

2. Problem-oriented functions solving described in the (Interim Report # 2, 2001 – Interim Report # 5, 2003) tasks (request translation, KSNet configuration, etc.);
3. Internal system functions to implementing internal behavior via a set of prepared scenarios. These functions allow finding different variables in the agent knowledge repository, store and read temporary results, preparing content of messages to other agent, etc.

3.2.2. Agents’ Interaction

Besides, the multiagent architecture of the system “KSNet” assumes availability of the following mechanisms:

- Mechanisms of agents’ registration by facilitator. Agents can be executed both in distributed environment (on different computers) and on the single computer. During a registration process information about the agent (ID, address, type, etc.) is sent to facilitator.
- Mechanisms of agent team formation. Two type of agent teams are used (Table 5):
 - peer-to-peer interaction assuming negotiation. This mechanism is best suited for applications that require fast, concurrent processing in order to come to a single solution without any conflict resolution. The problem with this type of infrastructure, however, is that it tends to be very brittle, and by leaving the burden of system level programming to agent developers, it becomes necessary to update each agent every time a new component is added to the network (Parrott et al., 2003).
 - mediated interaction assuming "master-slave" relation - federated agent network. It means that facilitator is used to mediate requests between agents. The advantage of this approach is that the addition of new or duplicate components is relatively easy, since only the facilitator's directory needs to be updated. The disadvantage is the potential processing bottleneck caused by requiring all messages to pass through a single facilitator.

Table 5. Agents' connectivity matrix. (P – peer-to-peer interaction, M – mediating interaction)

<div> <div></div> <div>Callee</div> </div>	Wrapper	Mediator	Facilitator	User Agent	Translation Agent	EA Agent	Configuration Agent	KF Agent	Monitoring Agent	OM Agent
Caller										
Wrapper		P	P		P					P
Mediator	P			P	P	P	P	P	P	
Facilitator		P								
User Agent		M/P	P							
Translation Agent										P
Expert Assistant (EA) Agent		M/P	P							
Configuration Agent	M/P	P	P					M/P		
KF Agent		P	P						P	
Monitoring Agent		M/P								
Ontology Management (OM) Agent	P				P				P	

3.2.3. Agent Knowledge Representation

A central question in reaching an effective deployment of multiagent system: how intelligent agents can be best designed and customized to meet users' individual information needs. The issue at stake concerns essentially one of adequate computational support. However, the motivations differ from those underlying linguistic frameworks for multiagent systems such as Actors Agha, 1986 and Agent-oriented Programming Shoham, 1993 as well as of behavior logic, either "reactive" (e.g. Brooks, 1991) or "deliberative" Bratman et al., 1988 or "hybrid" Kaelbling and Rosenschein, 1990.

It was decided to use the object-oriented constraint networks formalism for agents knowledge representation, what correlates with the knowledge representation formalism of the KSNet-approach.

Issues of the constraints application for agent-based systems were studied in a wide number of research efforts (Andreoli et al., 1994; Andreoli et al., 1996; Smith, 1980; Davis and Smith, 1983). The results show viability of this decision. For example:

1. constraints can be used to provide specifications on the possible values of variables, e.g., to implement predefined criteria for filtering and selecting information;
2. constraint-based knowledge brokers model exploits constraints to support knowledge-intensive tasks executed by agents negotiating in distributed environments as a form of distributed problem solving, it is capable of understanding and enacting both "requests" and "negations of requests", is self-sufficient in managing its own scope of exploration over a given information domain and is capable of knowledge reuse.
3. representing electronic information requires specific type of constraints (e.g., signed feature constraints), which can be used for number of specific issues of information management, such as interdependencies, thresholds, and reuse of information.

3.3. Constraint-Based Contract Net Protocol

Most multiagent systems require using agent negotiation models to operate. The negotiation models are based on the negotiation protocols defining basic rules so that when agents follow them, the system behaves as it supposed to. The main protocols including voting, bargaining, auctions, general equilibrium market mechanisms, coalition games, and contract net protocol (CNP) (Sandholm, 1993, Sandholm, 1999).

In the system "KSNet" the negotiation is required in two scenarios: (i) negotiation between configuration agent and wrappers during KSNet configuration and (ii) negotiation for expert knowledge conformation during distributed direct knowledge entry.

The main specifics of the KSNet-approach related to making a choice of the agent negotiation model were formulated as follows:

1. *Contribution*: the agents have to cooperate with each other to make the best contribution into the overall system's benefit – not into the agents' benefits;
2. *Task performance*: the main goal is to complete the task performance – not to get profit out of it;
3. *Mediating*: the agents operate in a decentralized community, however in all the negotiation processes there is an agent managing the negotiation process and making a final decision;
4. *Trust*: since the agents work in the same system they completely trust each other;
5. *Common terms*: since the agents work in the same system they share a common vocabulary and use common terms for communication.

Based on the analysis of the agents negotiation protocols and the above requirements to them, a CNP (Smith, 1980) was chosen as a basis for the negotiation model in the KSNet-approach. As it can be seen from Table 6 this protocol meets most of the requirements. In CNP the agent initiating negotiation process is called *manager*. Agents involved in the negotiation process by manager are called *contractors*.

Table 6. Comparison of negotiation protocols for KSNet approach

Protocols Criteria	Voting	Bargaining	Auctions	General Equilibrium Market Mechanisms	Coalition Games	Contract Nets
Contribution	<input checked="" type="checkbox"/> ¹	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/> / <input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Task performance	<input type="checkbox"/> / <input checked="" type="checkbox"/> ²	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/> / <input checked="" type="checkbox"/>
Mediating	<input type="checkbox"/> ³	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>
Trust	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>
Common terms	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>

¹ ☒ - supported

² ☐ / ☒ - weakly supported

³ ☐ - not supported

3.3.1. Conventional Contract Net Protocol

CNP is one of basic coordination strategies between agents in multiagent systems. It was originally introduced by Randall Davis and Reid G. Smith (Davis and Smith, 1983). The main features of this protocol are (i) *managers* (*initiators* in FIPA) divide tasks, (ii) *contractors* (*participants* in FIPA) bid, (iii) manager makes contract for the lowest bid, (iv) there is no negotiation of bids. The UML sequence diagram of FIPA-based CNP is presented in Figure 23. Since CNP is a basic protocol any particular multiagent system requires some modifications for CNP to be implemented (Payne, et. al., 2002, Sandholm and Lesser, 1995; Zhang, et. al., 2002). In the rest of this section the modifications made for original CNP required for a system “KSNet” (based on the KSNet-approach) are described.

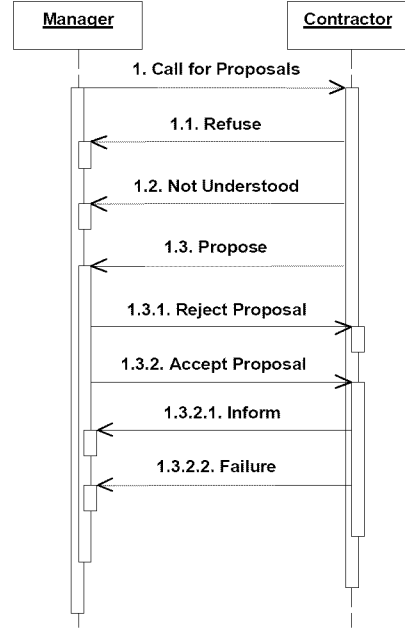


Figure 23: UML sequence diagram of FIPA-based contract net protocol

Improvements to the conventional CNP concern the formalism for agents’ knowledge representation and for communications between agents, and a scenario of the agents’ interaction described in the following two sections.

3.3.2. Constraint-Based Negotiation

Since the system “KSNet” uses object-oriented constraint networks for knowledge representation this formalism has been chosen for representation of agents’ knowledge and for agents’ communications. Two types of constraints are defined for agents: “local” and “global”. Each contractor agent deals with the local constraints describing its limitations, e.g., time of the task execution cannot be less than some value ($\text{time} \geq \text{time}_{\text{lower bound}}$). Manager also deals with the local constraints and the main purpose of these constraints is definition of requirements for the task execution, e.g., the task execution cannot last longer than some time ($\text{time} \leq \text{time}_{\text{upper bound}}$). Manager can also have objectives such as minimization of the task execution costs ($\text{costs} \rightarrow \text{min}$). Global constraints describe constraints defined by an agents’ community, e.g., resource limitations. The constraints can concern such parameters as time, costs, reliability, agents/resources availability/unavailability, network traffic, etc. For constraints processing the technology of constraint satisfaction / propagation is used.

Thus, a generic call for proposals from a manager to contractors has the following form:

Objective (optional)

Constraints (optional)

Content (required)

Objective is optional and used for meeting manager's constraints such as minimization of costs for a task processing. Constraints are also optional and used for a similar purpose, namely to meet requirements for a task execution. Content is a message itself including functions to be performed by contractors and other parameters.

Contractors' proposals besides a content part contain constraints corresponding to manager's objective and constraints. This will be illustrated in the example given at the end of this section. If contractors cannot meet the requirements of the manager they still can propose the closest possible parameters and it is up to the manager to decide whether to accept the proposal or not.

The negotiation process terminates when an acceptable solution is found or no improvement is achieved at the current iteration.

3.3.3. Modifications of Conventional Contract Net Protocol Interactions

Modification of the CNP-based negotiation model include introduction of additional messages and features required for KL related tasks, most important of which are presented in (Table 7) and illustrated in Figure 25 – Figure 27 via UML diagrams.

Table 7 represents agents roles in the two major scenarios of the system "KSNet". The first scenario of a KSNet configuration is executed as a part of the above step-by-step example of user request processing. The second scenario "Direct knowledge entry" (Interim Report # 4, 2002) consists of two subscenarios: execution and conformation. In these subscenarios expert assistant agents change their roles from "service provider" to "contractor".

Table 7. Changes in features of the conventional CNP required for the system "KSNet"

Protocols Feature	Conventional CNP	Modified CNP for the system "KSNet"
Iterative negotiation	-	the negotiation process can be repeated several times until acceptable solution is achieved
Conformation	-	concurrent conformation between manager and contractors
Available messages	fixed set of 8 messages (Figure 23)	flexible set: new messages specific for KF process and corresponding to FIPA Request and Confirm communicative acts, and message Clone (Figure 24) not corresponding to any FIPA communicative act are included (this message requests facilitator to create new instances of ("clone") mediators, user agents or expert assistant agents)
Participants roles	manager and contractors	manager and two types of contractors: (i) "classic" contractors negotiating proposals, and (ii) auxiliary service providers not negotiating but performing operations required for KL (e.g., ontology modification, user interfacing, etc.)
Role changing	-	agents can change their roles during a scenario

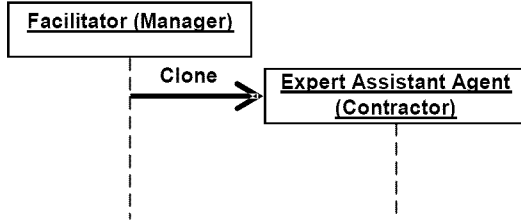


Figure 24: Example of new available messages

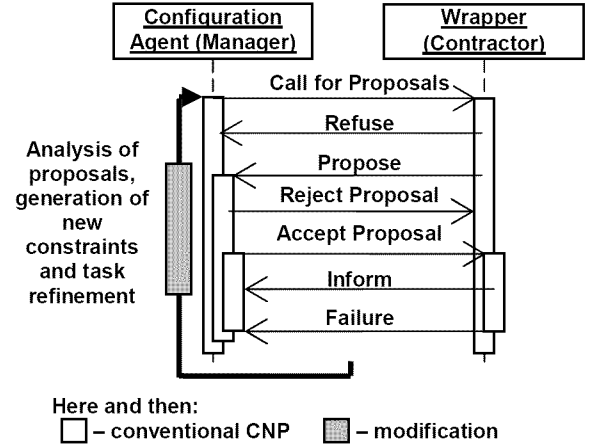


Figure 25: Example of the constraint-based iterative negotiation

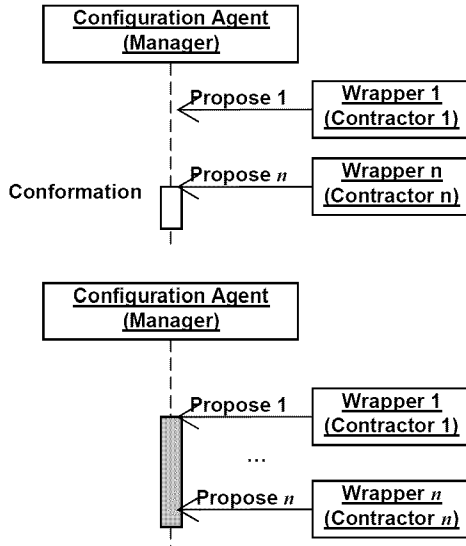


Figure 26: Example of conformation

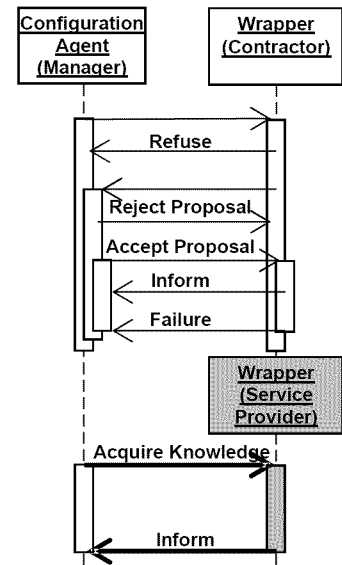


Figure 27: Example of new agents' roles and roles changing

3.4. Implementation of Agents Community

3.4.1. Multiagent Platform and Agent Operations

For agents community implementation it was necessary to choose a tool for prototype development. Based on the analysis of the multiagent systems development toolkits (e.g., FIPA-OS, 2002; Zeus, 2003; JADE, 2002; Gorodetski, et. al., 2001) the following requirements to them for implementation of KL technology were formulated: (i) availability of the source code, (ii) possibility of standard agent functions and features extension (introduction of new functions specific for KL technology), (iii) possibility of external functions calls (e.g., methods of registered COM/DCOM-objects, DLLs, etc.), (iv) possibility of configuration recovering after system faults, (v) availability of a name server storing information about registered agents, (vi) availability of message exchange mechanisms, (vii) availability of agent's knowledge repository creation and modification for storage of agent's knowledge, operation scenarios, and behavior rules, (viii) availability of agent's “pro-activity” functions.

For the research prototype implementation the MAS DK environment developed in SPIIRAS (Gorodetski, et. al., 2001) has been chosen as a toolkit for agents interaction modeling. Agents' scenarios use finite state automata starting when one or more predefined conditions are met (an agent receives a message from another agent or from the system). Each finite state automaton is implemented using the internal MAS DK language. It is represented by a set of conditional statements, messages to other agents, and agent's functions. Conditional statements are used to check error codes of problem-oriented functions, to compare values of different variables stored in the agents' repository, etc.

The agent's identification component is presented in (Figure 28). The agent's functional component is presented in Figure 29.

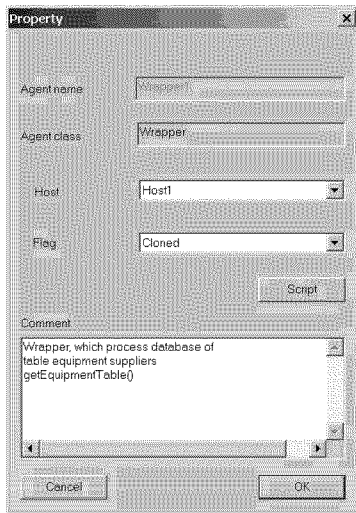


Figure 28. An example of agent's identification component

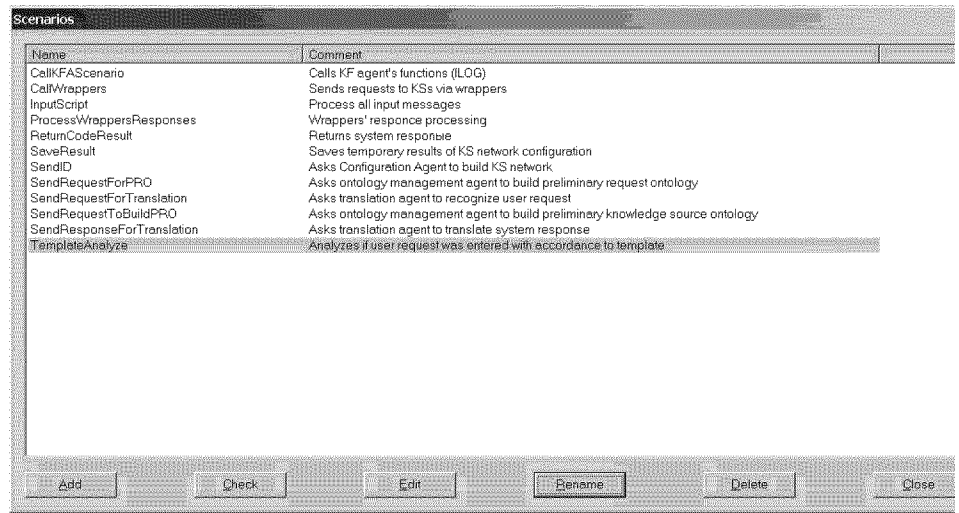


Figure 29. An example of agent's functional component

An example of developed scenario of the user request processing is presented in Figure 30. It begins with “*ReceiveRequest*” function and ends with “*ReturnResponse*” and “*Free*” functions.

Agents' functions were extended using Microsoft Visual C++, Microsoft Access XP, Microsoft Access ODBC drivers, constraint satisfaction/propagation technology ILOG (Configurator 2.0, Solver 5.0, Concert Technology 1.0, ILOG Dispatcher 3.2) (ILOG, 2003), lexical reference system WordNet (WordNet, 2003), etc. It was motivated by the following reasons: (i) ILOG is a world leader in constraint satisfaction/propagation technologies and optimization algorithms that would enable a very efficient processing of large amounts of constraints, (ii) ODBC allows utilizing unified functions for accessing relational databases of different database management systems and allows easy modification of the DBMS used in the prototype, (iii) Microsoft Visual Studio 6.0 (Visual C++) allows accessing ILOG features, databases, writing efficient programs, and (iv) Microsoft Access XP allows rapid database design and creating simple database applications. Some intermediate and auxiliary forms for data preparation and results visualization were designed using Microsoft Access. For representation and interchange of agent's messages XML was used. Due to a large number of tools working with this format and specifications describing it the application of XML was useful and convenient.

Some members of agents community are located on the same host (server) and use a shared directory for information exchange. They write auxiliary information to temporary files, read and modify them. Using developed mechanisms of file naming each agent knows which file it has to use for its work. Wrappers are distributed on different computers to provide access to KSs. Facilitator stores information about each agent location.

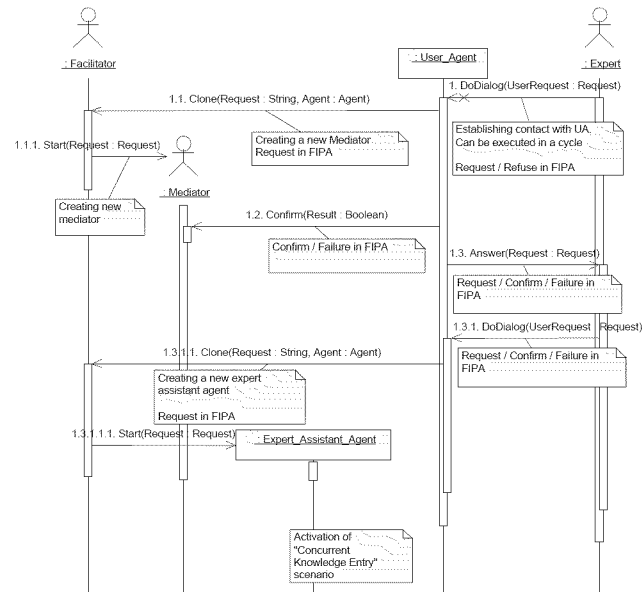


Figure 31. Knowledge entry process initialization by expert

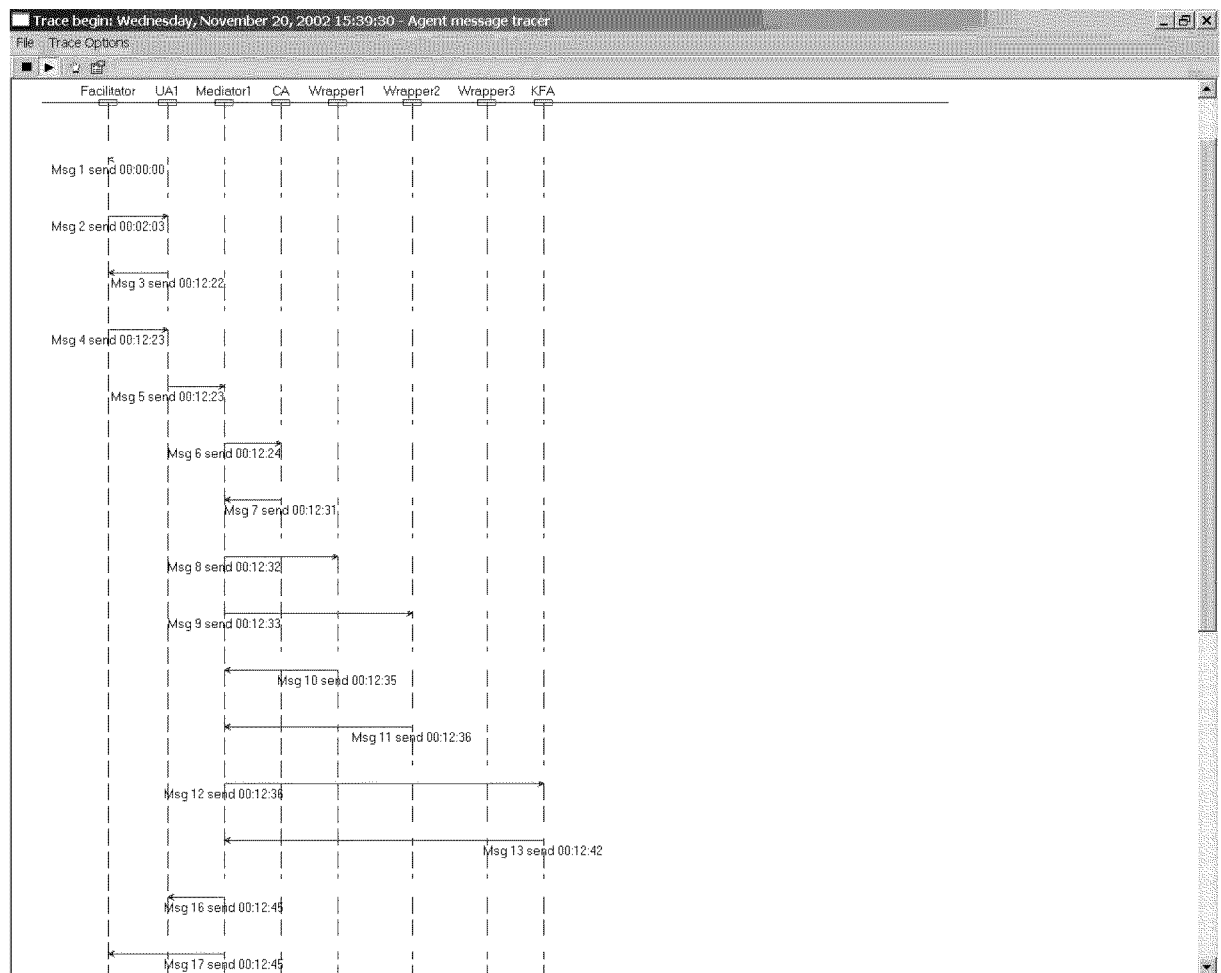


Figure 32. An example of agents' interaction

3.5. Structure of the System “KSNet” Research Prototype

The client–server architecture of the research prototype of the system “KSNet” is presented in Figure 33. It was chosen in accordance with the following reasons: (i) minimization of requirements to user computers (web-based application allows user to have only HTML-compatible web-browser and access to the Internet), (ii) requirement of processing large amounts of information received from distributed KSs on the central (server) computer, (iii) a necessity to have an access to ILOG key server located in the local network, and (iv) specifics of the agents community implementation.

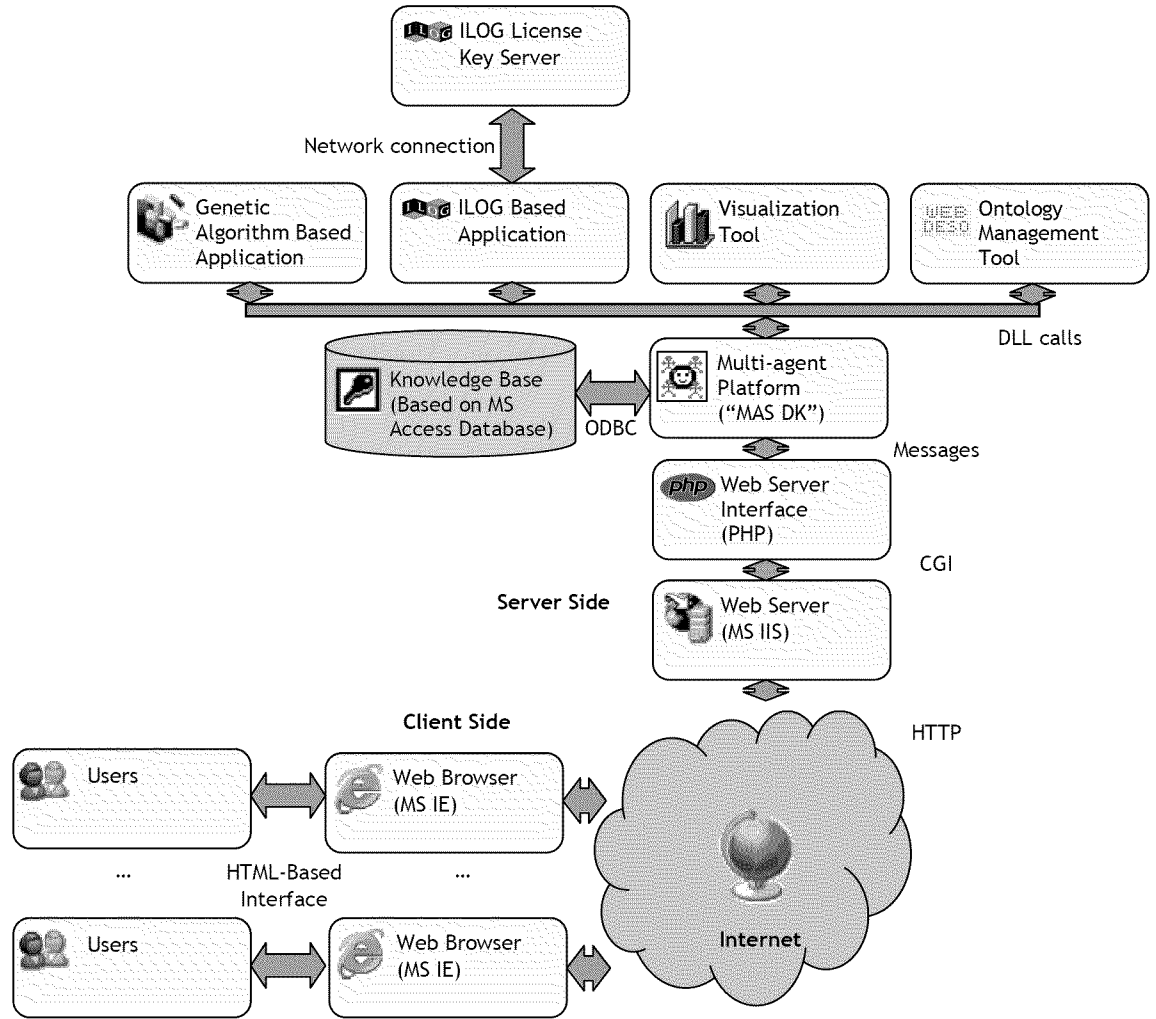


Figure 33. Prototype architecture

The client interface is implemented as HTML pages with Java-scripts presented to users via MS Internet Explorer. The server part is implemented using PHP v.4.2.3, MS Access XP, MS Access ODBC drivers, MS Visual Studio 6.0 (Visual C++, Visual FoxPro) and constraint satisfaction/propagation technology ILOG (Configurator 2.0, Solver 5.0, Concert Technology 1.0, Dispatcher 3.2). The group decision support system “MultiExpert” (Smirnov et al., 1997) is used for the knowledge map creation. Agents community (Figure 34) developed using MAS DK toolkit.

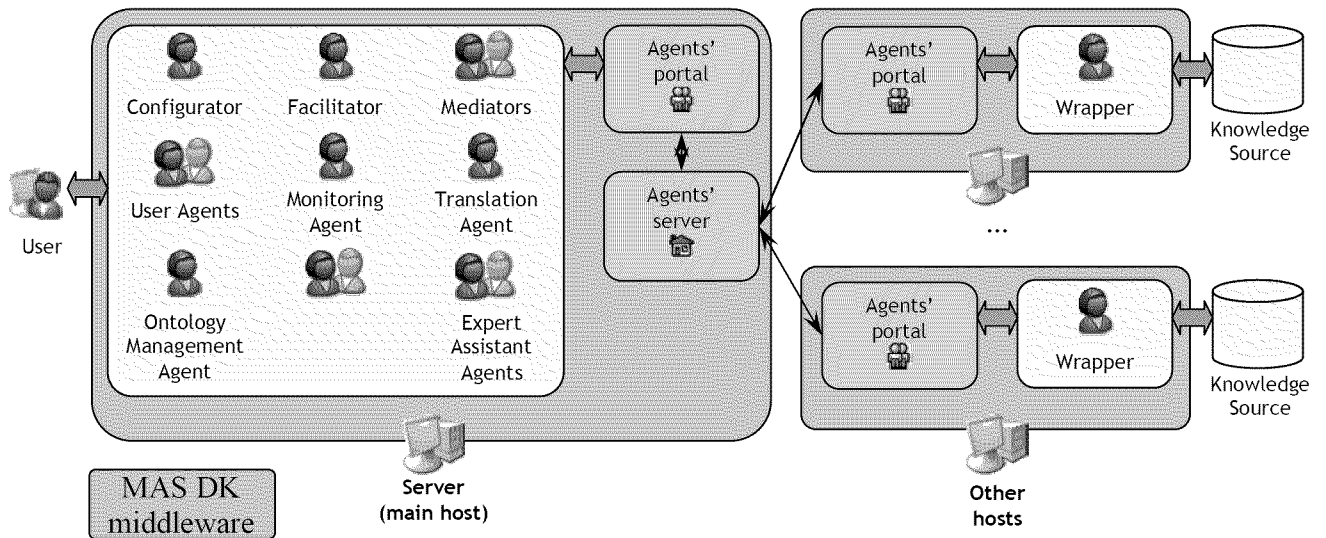


Figure 34. Agents community of the system “KSNet”

In accordance with up-to-date technologies and standards the information kernel for KL was built as shown in Figure 35. As it was described above the knowledge in the system is represented by an aggregate of interrelated classes, their attributes, attributes' domains, and relations between them. An object scheme for working with the knowledge and a database structure for its internal storage are designed based on this notation. An access to the database is performed via ODBC as a standard data access mechanism under MS Windows. Remote access to the stored knowledge is performed via common HTTP Internet protocol. Knowledge is represented by either interactive HTML+VRML JavaScript enabled pages for users or a format based on DAML+OIL for knowledge-based tools.

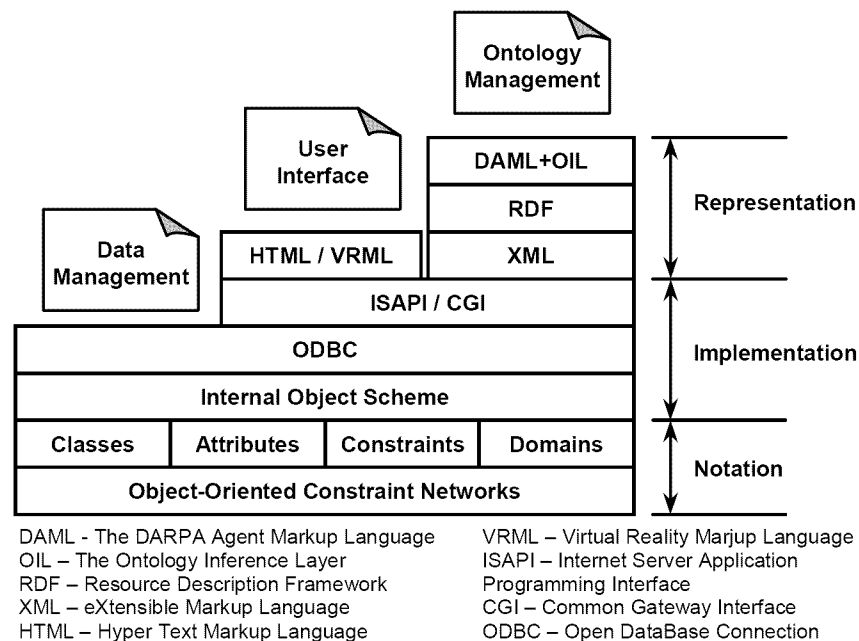


Figure 35. Standards of knowledge logistics information kernel

A list of main functions of the system “KSNet” users and agents interacting with users to perform these functions are given in Table 8. The users are divided into two groups: (i) users serving the system and (ii) users served by the system (knowledge customers).

At the early stage of the system’s operation, knowledge consumers and experts pass the registration procedure. This process assumes entry of initial user information by filling a form or a questionnaire, generating the user profile, granting user rights. Later the user passes the authentication (user rights verification) process only.

Table 8. The system “KSNet” users

User Types	Main Functions	Agents
Administrator	<ul style="list-style-type: none"> ➤ service table processing (archiving, indexing, etc.); ➤ backup creation; ➤ system diagnostics; ➤ diagnostics reports processing; ➤ user rights management. 	<ul style="list-style-type: none"> ➤ Monitoring agent
Experts	<ul style="list-style-type: none"> ➤ application domain studying, ➤ KSO creation, ➤ AO creation, ➤ search for KSs, ➤ alternative KSs ranking (regarding the domain). ➤ new knowledge entry, ➤ alternative KSs ranking (regarding the stored requests). 	<ul style="list-style-type: none"> ➤ Monitoring agent ➤ Mediator ➤ Facilitator ➤ Expert assistant agent ➤ User agent
Knowledge consumers	<ul style="list-style-type: none"> ➤ input of requests (“user requests”). 	<ul style="list-style-type: none"> ➤ User agent ➤ Mediator ➤ Monitoring agent ➤ Facilitator ➤ Translation agent ➤ Ontology management agent ➤ Wrapper ➤ Configuration agent ➤ KF agent
Software engineers	<ul style="list-style-type: none"> ➤ adaptation, creation, and connection of methods for task solving, parsing and translation of attribute values obtained from KSs; ➤ connection of new KSs and KS types. 	<ul style="list-style-type: none"> ➤ All agent types
Ontology engineers	<ul style="list-style-type: none"> ➤ AO creation; ➤ preliminary KSO and KSO creation; ➤ request ontology creation; ➤ modification of ontologies stored in OL. 	<ul style="list-style-type: none"> ➤ User agent ➤ Ontology management agent ➤ Monitoring agent ➤ Facilitator ➤ Mediator
Knowledge engineers	<ul style="list-style-type: none"> ➤ internal knowledge base validation; ➤ search for missing knowledge. 	<ul style="list-style-type: none"> ➤ User agent ➤ Expert assistant agent ➤ Mediator ➤ Monitoring agent ➤ Facilitator ➤ Translation agent ➤ Ontology management agent

Agents support (i) work of ontology engineers: help them to import and build ontologies, (ii) the process of direct knowledge entry by knowledge engineers into the internal knowledge base, (iii) the process of alternative KSs ranking by experts, and (iv) the process of knowledge customer requests processing. They provide administrator the results of KS monitoring. The software engineers interact with agents indirectly – they develop and attach methods which are used by the agents and take part in the development of the wrappers for new types of KSs.

The interface forms for all users of the system “KSNet” were developed for the software prototype. Two types of interface forms were designed for knowledge customers: (i) request input in an arbitrary form and (ii) problem-oriented structured request input called *user request templates*. When the system “KSNet” starts working it does not have any request templates. Users input requests in an arbitrary form and the system accumulates information about users’ requests and interests.

Implementation of problem-oriented agents and description of experiments performed by the developed research prototype are presented in the next two sections.

4. TECHNOLOGIES AND TECHNIQUES OF PROBLEM-ORIENTED AGENTS

This section describes technologies and techniques implemented in problem-oriented agents.

4.1. Configuration Agent: Configuration Management of Knowledge Source Network

One of the tasks the Configuration Agent has to solve is the task of efficient KSs choice. For this task a number of solution techniques were tried and a Genetic Algorithm (GA) was chosen as a probabilistic approach to pseudo optimal solutions search. It suited best for the task of the enumeration nature. Among other techniques such as k-nearest neighbor (this technique was computationally intensive for large data sets) and decision tree (this technique did not provide satisfactory reduction of the decision space) worth mentioning.

Initially a random set of solutions is generated, then solutions are estimated; the set is sorted according to the chosen criteria, and mutation mechanism is applied to the best solution to generate new solutions. The newly generated solutions are sorted, and the new iteration is performed. The process is stopped after a predefined number of iterations.

User request processing assumes an acquisition of knowledge from KSs. Sometimes several KSs can be used for acquisition of the same knowledge. As a result a task of choosing appropriate sources arises. The goal of this task is a selection of KSs to be used for the user request processing in a most efficient way according to predefined criteria such as costs, time, and reliability. The mathematical model presented below defines components of the complex elements of the task (AO, KS and KS ontologies, knowledge map, user request and user request ontologies). Then it sets dependencies between the elements and defines the objective function.

AO A contains some ontology elements (OE – $\{a_j\}$), i.e. classes O , attributes Q , domains D , and constraints C of the application domain.

$$A = (O, Q, D, C) = \{a_j\}, j = 1, \dots, n \quad (1)$$

where n is the number of OEs.

KS S_i contains some OEs $\{s_{it}\}$ at a time instant t . Besides OEs, KS contains instances (information content I), i.e. it constitutes a constraint network $CNet(S_i)$:

$$\begin{aligned} CNet(S_{it}) &= \\ &= (O(S_{it}), Q(S_{it}), D(S_{it}), C(S_{it}), I(S_{it})) = \\ &= \{s_{it}\}, i = 1, \dots, m, t = 1, \dots, T, l = 1, \dots, p_i \end{aligned} \quad (2)$$

where m is the number of KSs in the system, T is the system life time, and p_i is the number of OEs of KS_i .

Knowledge map associates OEs of KSs with those of AO at a time instant t . Such association is denoted by a symbol “ \rightarrow ”, and a statement “ $OE a_j$ is associated with $KS S_{it}$ ” is denoted by $(a_j \rightarrow S_{it})$:

$$KM_t = \{(a_j \rightarrow S_{it})\}, a_j \in A \quad (3)$$

It is considered that for each KS its parameters such as costs, availability, access time, on-line schedule, etc. are known. KS ontology will be defined as an association of KS’ elements with AO’s elements:

$$A(S_{it}) = \{(a_j \rightarrow s_{it})\} \quad (4)$$

When a user request R is received by the system it is decomposed into a set of subrequests r_k , which then are associated with the AO's OEs (i.e. translated into the system's terms). This association is contained in the request ontology $A(R)$. When these operations are completed the request translated and decomposed into subrequests associated with the AO's OEs will be obtained (denoted by R'):

$$R = \{r_k\} \quad (5)$$

$$A(R) = \{(r_k \rightarrow a_j), r_k \in R, a_j \in A\} \quad (6)$$

$$R' = \{a_j\}, \forall a_j \in R' : \exists (r_k \rightarrow a_j) \in A(R) \quad (7)$$

When the operations above are completed a set of feasible decisions of the task Dec_R can be written as:

$$Dec_R = \{dec_R\}, dec_R = \{(r_k \rightarrow s_{lit})\} \quad (8)$$

Costs $Cost$ or time $Time$ required for request processing can be used as criteria of the decision's effectiveness:

$$Cost = f_{Cost}(dec_R) = \sum_{s_{jik} \in dec_R} f_{Cost}(s_{jik}) \quad (9)$$

$$Time = f_{Time}(dec_R) \quad (10)$$

Also, an overall index of effectiveness Eff including estimations of both costs and time can be considered (multicriteria optimization). For instance, normalized values of cost and time (superscript N) functions can be summarized using weights w_{Cost} and w_{Time} :

$$\begin{aligned} Eff &= f_{Eff}(dec_R) = \\ &= f'_{Eff}(f_{Cost}(dec_R), f_{Time}(dec_R)) = \\ &= w_{Cost} \cdot f_{Cost}^N(dec_R) + w_{Time} \cdot f_{Time}^N(dec_R), \\ w_{Cost} + w_{Time} &= 1 \end{aligned} \quad (11)$$

Decision is considered effective (denoted by dec_R^{eff}) if the value of goal function, e.g., (11), is minimal with the constraints (1 - 8) being true:

$$\begin{aligned} dec_R^{eff} &\in Dec_R, \\ \forall dec_R \in Dec_R, f_{Eff}(dec_R^{eff}) &\leq f_{Eff}(dec_R) \end{aligned} \quad (12)$$

To solve this task an application of the genetic algorithm (GA) is proposed. GA is proved to be efficient for tasks of the enumerative nature but it has not yet been applied to the problem of KSs choice. A feasible static decision dec_R represents a chromosome and has the following structure:

$$dec_R = \{dec_{k,i}^R\} \quad (15)$$

where each $dec_{k,i}^R$ is a Boolean variable equal to 1 if KS_i is used for obtaining OE_k or to 0 otherwise. Hence, dec_R represents a binary matrix (Figure 36), whose rows are considered as genes for GA.

	S_1	...	S_i	...	S_m
a_1	$dec_{1,1}^R$...	$dec_{1,i}^R$...	$dec_{1,m}^R$
...
a_k	$dec_{k,1}^R$...	$dec_{k,i}^R$...	$dec_{k,m}^R$
...
a_n	$dec_{n,1}^R$...	$dec_{n,i}^R$...	$dec_{n,m}^R$

Figure 36. Structure of a feasible decision used in GA

To investigate an efficiency of GA applied to the KSs choice, a set of experiments with a basic GA for tasks of different dimensions have been performed, with KSs' parameters and knowledge maps being randomly generated. The results indicate that GA applied to the KSs choice behaves as expected: the number of required calculations for obtaining a quasi-efficient decision using even basic non-optimized GA is smaller than that in the method of exhaustive search. Figure 37 represents the ratio of calculations number for the method of exhaustive search to that for the GA. As the task dimension grows this improvement grows nonlinearly. This proves that application of GA to the KSs choice is justified.

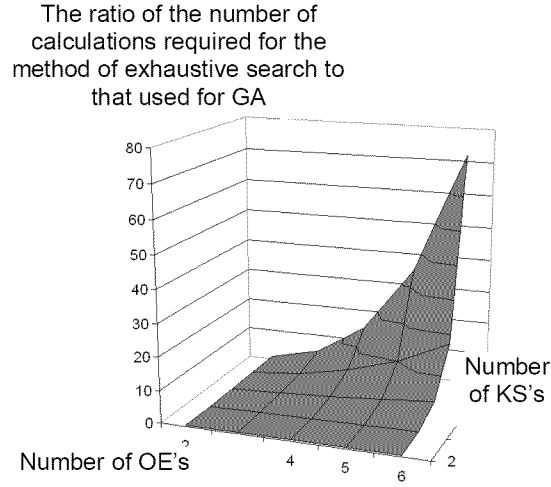


Figure 37. Efficiency improvement due to GA application

4.2. Monitoring Agent: Knowledge Indexing and Web Services

The *monitoring agent* provides a set of functions for diagnostics of the *internal knowledge base*, external KSs, and extern Web-services.

When the system starts the facilitator sends a message to a monitoring agent. The monitoring agent starts working, checks its own schedule and waits for messages from other agents or from the system. Figure 38 represents an example of XML message received by monitoring agent from expert assistant agent. This message contains information about capacity of supplier "OPTIWAY TECHNOLOGY INC".

4.2.1. Knowledge Map Maintenance

Knowledge map is a component of the repository developed for knowledge indexing. It contains (i) KS characteristics, (ii) information about relations between pairs of classes and their attributes from AO and KSs, and (iii) information about alternative KSs.


```

<?xml version="1.0" standalone="yes"?>
<!--The system "KSNet", SPIIRAS, 2002-->
<Hospital_Bed>
  <Supplier>    <!--Beginning of the information block about supplier -->
    <name>OPTIWAY TECHNOLOGY INC</name><!--Name of supplier -->
    <model>99803</model>          <!--Model of bed -->
    <quantity>7</quantity>        <!--Quantity of beds -->
    <price>22</price>             <!--Price of bed -->
    <picture_filename>4.jpg</picture_filename><!--Picture of bed -->
    <lead_time>2</lead_time>      <!--Lead time -->
  </Supplier> <!--End of the information block about supplier -->
</Hospital_Bed>

```

Figure 38. An example of XML message received by monitoring agent

The monitoring agent provides a set of functions for diagnostics of the internal knowledge base and external KSs:

- Checking KSs for (i) availability, and (ii) changes. Modifying the knowledge repository depending on the returned code.
- Checking the knowledge repository for consistency. Modifying the knowledge repository or performing other actions depending on the returned code.
- Receiving a request for new knowledge (entered or generated as a result of KF) addition to the internal knowledge base. Adding new knowledge to the internal knowledge base and performing appropriate changes in the knowledge map. It can collaborate with knowledge engineers when it cannot perform assigned to it operations without assistance.
- Asking the facilitator for cloning a new mediator for support of alternative KS ranking.
- Defining a list of alternative KSs with their parameters.
- Preparing a task of alternative KSs ranking for a group of experts (forming a group of experts, preparing required information, asking for the mediator, sending message to the new mediator for interacting with the appropriate expert assistant agents).

Scenario of the alternative KS ranking supported by multiagent architecture is presented below (Figure 39).

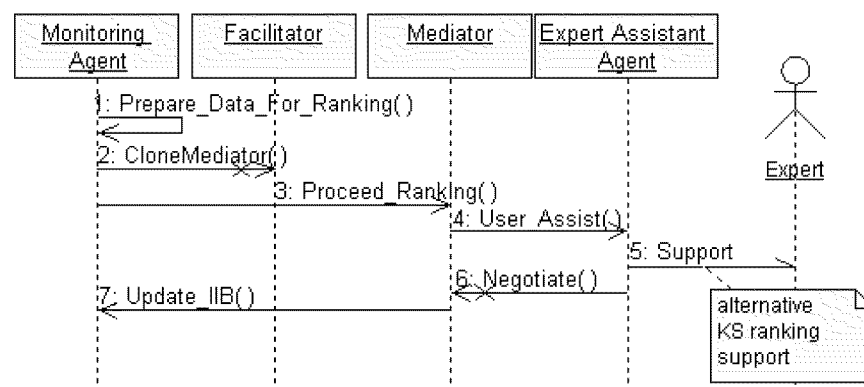


Figure 39. Sequence diagram of KSs ranking

1. *Ontology Engineer* assigns the task of alternative KSs ranking to the *monitoring agent*. The *monitoring agent* defines a set of alternative KSs for ranking, forms a group of experts and prepares tasks for the experts.

2. The *monitoring agent* asked the *facilitator* for a *mediator*, which will trace this task. The *facilitator* clones the *mediator* and assigns the task to it.
3. The *monitoring agent* prepares messages for the *expert assistant agents* and passes them to the *mediator*.
4. The *mediator* passes messages for the *expert assistant agents*.
5. *Experts* perform alternative KS ranking interacting with the *expert assistant agents*.
6. The *expert assistant agents* pass the results of the work to the *mediator*. The *mediator* performs conformation of the local expert opinions negotiating with them and returns the results to the *monitoring agent*.
7. The *monitoring agent* performs appropriate changes in the *internal knowledge base* and the *knowledge map*.

4.2.2. Web-Service Support

A networked organization environment can be characterized as an open service system (i) in which users, units and other resources come and go on a continual basis, and (ii) in which entities provide services to each other under various forms of a contract (or agreement). The key components of service oriented architecture are as follows: *service owners* that offer *services* to *service consumers* under particular *contracts*. Each owner-consumer interaction takes place in a given *marketplace* whose *rules* are set by *market owner*. The service owners and service providers interact with each other in a particular *environment context*. The open service oriented model applied to the system "KSNet" looks as shown in (Figure 40). The system acts as a service provider for knowledge customer services and at the same time as a service requestor for knowledge suppliers (OKBC-compatible knowledge representation systems).

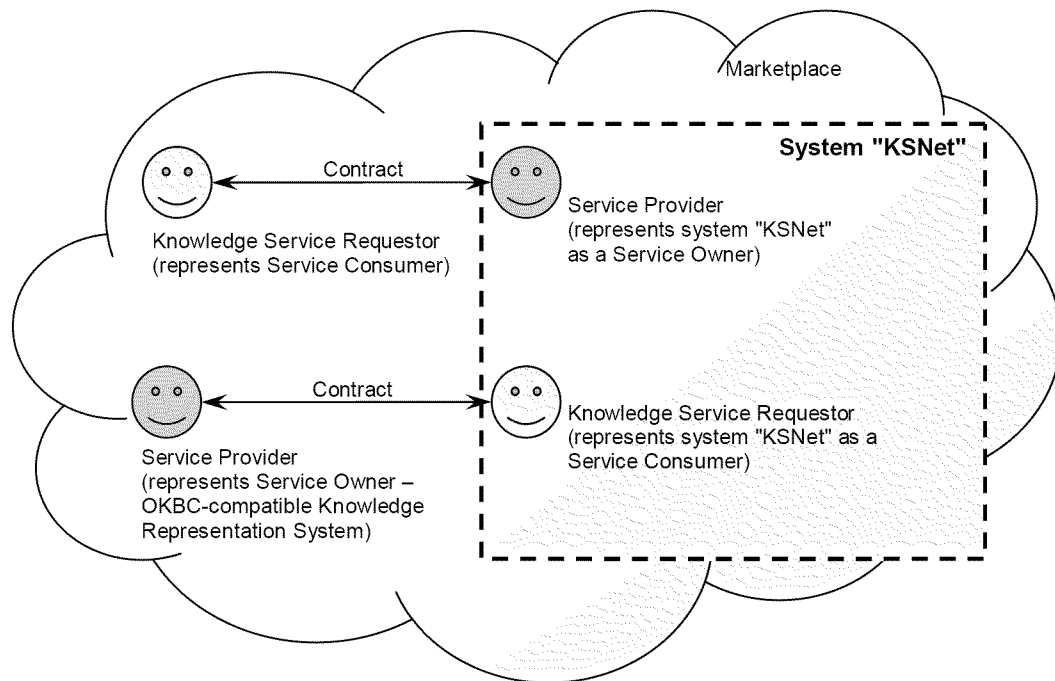


Figure 40. Open service oriented model of networked organization (adapted from Roure, et. al., 2003)

In order to implement this integration a number of problems have to be resolved. The most important of them include (The Globus Project Tutorial, 2003):

- Protocol deficiencies, e.g. heterogeneous basis: HTTP, LDAP, FTP; no standard means of invocation, notification, error propagation, authorization, termination, etc.;

- Significant missing functionality, e.g. databases, sensors, instruments, workflow, etc.; virtualization of end systems (hosting environments).

As a possible technological basis resolving these problems the technologies of open services (Web-services and Grid-services) are proposed.

The conceptual scenarios of both Web and Grid services are shown in (Figure 41, Figure 42). A service requester sends a request for an appropriate service ("Find Service") to the UDDI Registry (for Web services) or to the Factory (for Grids). The Registry returns Grid Service Reference (GSR) and Grid Service Handle (GSH) of the appropriate service to the requestor. In Grids the Factory first creates a new instance of the required service and only then returns GSR and GSH of the new service to the requestor. At this point the scenarios for both Grid and Web services are the same: requestor sends a request to the service and receives the result from it. When this operation ends in Grids the service terminates.

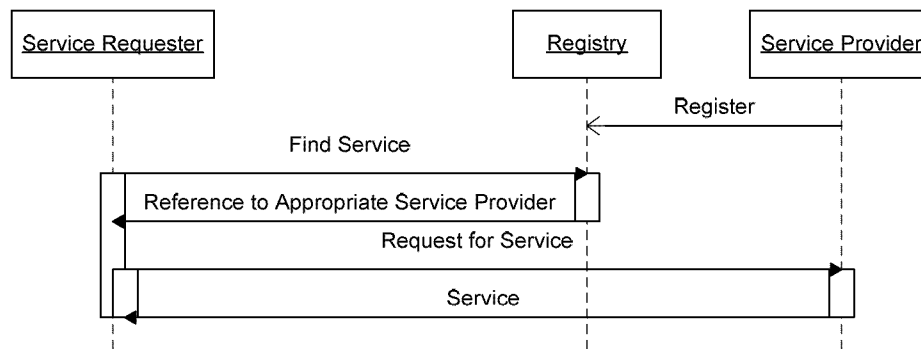


Figure 41. Conceptual Web service scenario

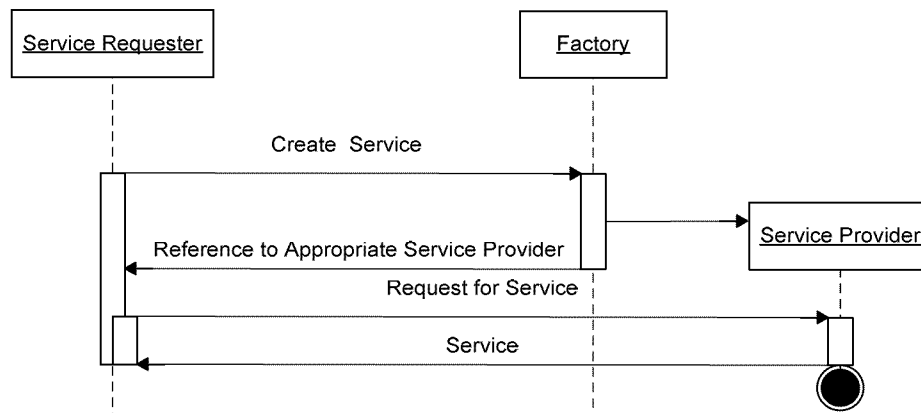


Figure 42. Conceptual Grid service scenario

In the system "KSNet" the Grid-based open service model is used. The proposed scenario is presented in (Figure 43). The main specific is that the service passes the request into the system where it goes through all the stages of the request processing scenario. When an answer for the request is found it gets to the service and then it is passed to the requestor.

Monitoring agent via Web service interface looks for requests on obtaining new knowledge. After it has got the request it checks it and works in accordance with the main scenario. Monitoring agent sends reply to the service. Web service passes it to the requestor (Figure 43).

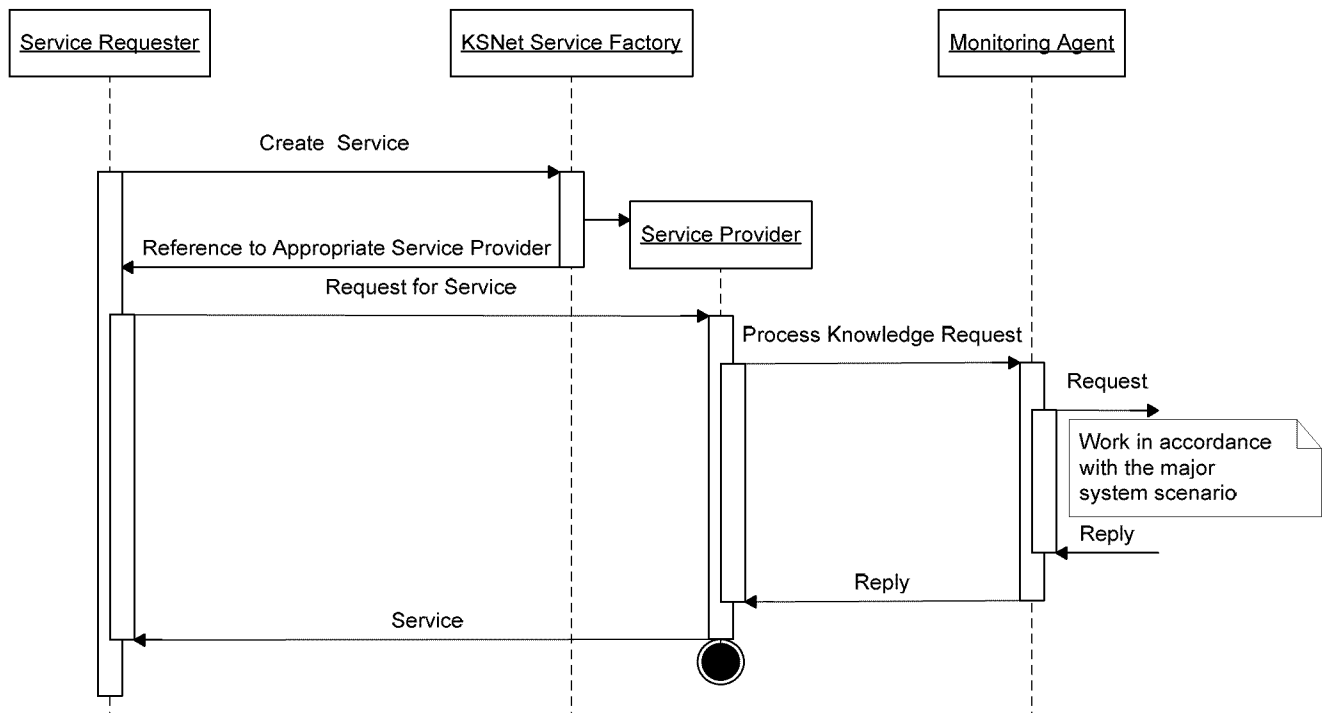


Figure 43. Knowledge Grid service scenario for the system "KSNet"

4.3. Knowledge Fusion Agent: On-the-fly Constraint Satisfaction Mechanism

KF agent performs KF based on AO, user request ontology, knowledge acquired from KSs and utilizes such technologies as constraint satisfaction/propagation and optimization. In the prototype of the system "KSNet" the features of ILOG Configurator, Solver and Dispatcher are used. The agent's activity is defined by a content of input messages.

Figure 44 presents the input messages containing parameters of hospital configuration task. These parameters define type and quantity of required goods, i.e. it is required 14 hospital beds and 7 operating tables for hospital functioning.

The user request defines both the problem statement and what data has to be retrieved from OL and from KSs. Thereby the problem statement is changed from one request to another. The novel "on-the-fly" compilation mechanism in combination with ILOG (ILOG, 2003) is proposed to solve these varying problems. In a rough outline this novel "on-the-fly" compilation mechanism is based on the following concepts (Figure 45):

- a pre-processed user request defines (1) which ontologies are to be extracted from OL, and (2) which KSs are to be used;
- C++ code is generated on the basis of information extracted from (1) the user request (goal, goal objects, etc.), (2) appropriate ontologies (classes, attributes, and constraints), and (3) suitable KSs;
- the compilation is performed in an environment of the prepared in advance C++ project;
- failed compilations/executions do not fail the system work in whole; an appropriate message for the user is generated.

```

<?xml version="1.0" standalone="yes"?>
<!--KSNet SPIIRAS-->
<Hospital>
  <destination_id>8</destination_id><!-- Destination point ID      -->

  <!-- Information about one of required hospital components (bed) -->
  <GoodsNeed>
    <type>Hospital_Bed</type>      <!-- Type of component: Hospital Bed -->
    <quantity>14</quantity>      <!-- Required quantity: 14      -->
  </GoodsNeed>

  <!-- Information about another required hospital component (table) -->
  <GoodsNeed>
    <type>Operating_Table</type> <!-- Type: Operating Table      -->
    <quantity>7</quantity>      <!-- Required quantity: 7      -->
  </GoodsNeed>
</Hospital>

```

Figure 44. An example of XML file with task parameters

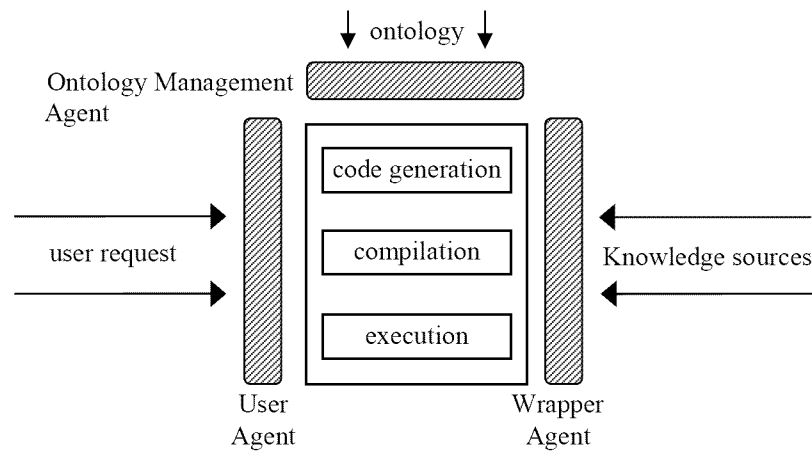


Figure 45. Solver “on-the-fly” compilation mechanism

The KF agent is directly coupled with the Solver. The described earlier mechanism of “on-the-fly” compilation assumes that the KF agent gathers required data from other the system “KSNet” agents (the translation agent, the ontology management agent, the configuration agent and the wrappers), generates a solver (performs the “on-the-fly” compilation), and launches the solver to generate a solution. In more detailed fashion this particular task performed by the KF agent requires (Figure 46):

- a pre-processed user request prepared by the translation agent;
- an appropriate ontology demanded from the OL found and extracted by the ontology management agent;
- knowledge received from KSs by wrappers.

ILOG Solver has been chosen as a constraint solver because it has the following valuable features:

- it is based around some C++ libraries;
- it has functionalities for managing constraints;
- it has functionalities to program:
 - user-defined solving process;
 - user-defined constraints propagation mechanisms.

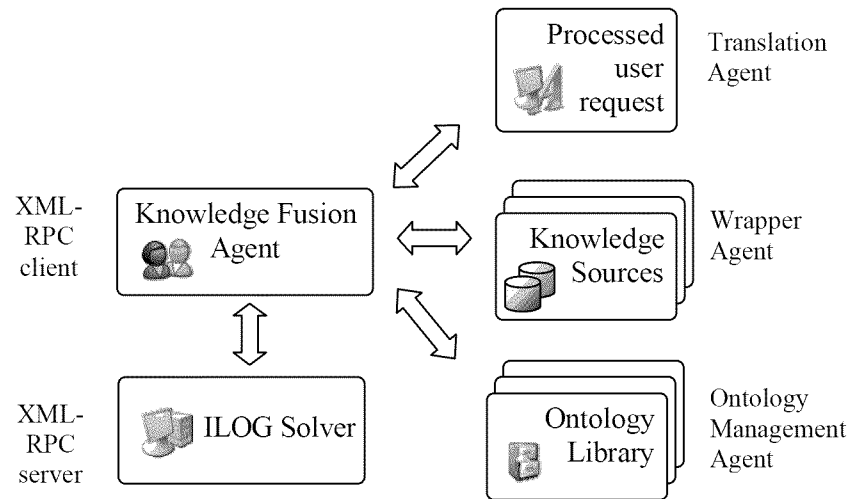


Figure 46. KF agent activity

The KF agent and ILOG Solver interact via the XML-RPC (XML-RPC, 2003) protocol (remote procedure calling via HTTP as a transport protocol and XML as an encoding language). XML-RPC is designed to be as simple as possible, while allowing complex data structures to be transmitted, processed and returned.

XmlRpc++ library (XmlRpc++, 2003) is used as an implementation of the XmlRpc protocol written in C++. XmlRpc++ is designed to make it easy to incorporate XmlRpc client and server support into C++ applications. It has the following features:

- Easy – this library is easy to incorporate into C++ applications. No other libraries are required, except system's socket libraries. Simple XML parsing and HTTP support are built in;
- Fast – all input/output operations are non-blocking, so a slow client or network will not slow down the server;
- Portable – written in standard C++ to the POSIX⁵ and Windows sockets APIs;
- Free – this library is released under the GNU LGPL (GNU, 2003).

The KF agent performs KF based on AO and knowledge acquired from KSs. The implementation of the KF agent uses such fundamental ideas of programming languages as object-oriented approach and constraint programming. ILOG Configurator (ILOG, 2003) was chosen as a generic tool for object-oriented constraint programming. It provides a library of re-usable and maintainable C++ classes. These classes define objects in the application domain in a natural and intuitively way so that it is possible clearly distinguish the problem *representation* from the problem *resolution*. Therefore, if a problem statement changes then it is not necessary to rewrite the entire code as in case of “pure” C++. In the given case the problem statement is defined by ontology elements retrieved from OL, therefore the problem of minimal code modification, fast and error-free is the important task here.

The essence of the proposed on-the-fly compilation mechanism is to write the AO elements (classes, attributes, constraints) to a C++ file directly. The KF agent creates a C++ file based on these data and inserts program source code to a program (Microsoft Visual Studio project) prepared in advance. The program is compiled in order to create an executable file in the form of dynamic-link library (DLL). After that the KF agent calls a

⁵ POSIX threads — the current standard for threads is ISO/IEC 9945-1:1996 Information Technology — Portable Operating System Interface (POSIX) — Part 1: System Application: Program Interface (API) [C Language] (<http://standards.ieee.org/catalog>).

function from DLL to solve the task. The UML sequence diagram (Figure 47) shows the KF agent's scenario at the stages of knowledge obtaining, solver compilation and execution.

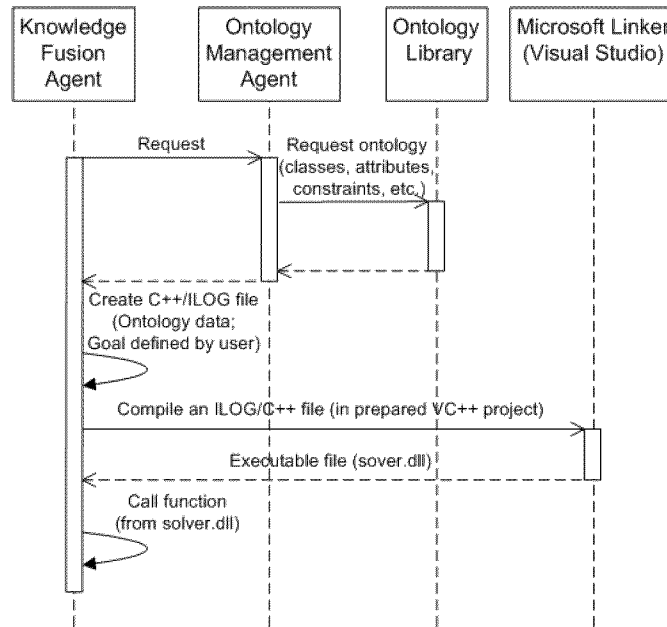


Figure 47. The UML sequence diagram shows the KF agent scenario

The KF agent uses the mentioned technique of dynamic code generation (with ILOG Configurator commands embedded) to produce a solution set satisfying requirements of the user request and AO elements (classes, attributes, constraints, etc.).

The generated code (C++ file) consists of several parts:

- The ontology management agent passes a part of the program based on data from the OL;
- The wrapper passes a part of the program using local/remote KSs;
- The KF agent generates a part of the program based on user request processing as well as user requirements;
- The predefined part of code (unchangeable): an algorithm and strategy definition and an automatic answer generation.

Thus the C++ file is created on the basis of a special template (Figure 48). This template allows researchers and developers to comprehend and realize in more explicit and well-defined form:

- what information is needed to solve task;
- which KSs are required;
- which agent is responsible for delivering particular specified information block.

Therefore instead of solving one complete problem the solver has to solve easier solvable subproblems one after another (Figure 49). Since the subproblems are parts of the complete problem, therefore there are “part-of” relation between the complete problem and the subproblems. The task statement of the complete problem contains global constraints. The task statement for each subproblem is a set of local constraints. These constraints are passed to the ILOG Solver. Because of hardware restriction only one subproblem can be computed via the “on-the-fly” compilation mechanism in combination with ILOG at a moment of experiment. Hence there is a queue that is subproblems ordering, so that first subproblem offered to the solver will be solved first.

```

// include standard C++ library

int getHospILOG(void)
{
    // variables definition

    // 1. Type and attribute declarations
    //class=Hospital
    IloComponentType type_7 = cat.addType ("type_1");

    // Hierarchical constraints
    //Classes: parent=Hospital; child=Goods
    IloPort port_7_8 = type_7.addPort (IloHasPart, type_7, "port_7_8",1);

    // Functional constraints
    // Supplier.add (SupplierProg == 0);
    type_9.add (attr_9_1 == 0);

    // 2. Objects (instances of classes)
    // type_7 is class "Hospital"
    IloComponent object_7_1 = request.makeInstance (type_7, "object_7_1");

    // 3.Goal definition (generated by script based on user requirements)

    // 4. Algorithm & Strategy Definition; Solving

    // 5. Automatic answer generation (print to screen)
    // (the attributes of each component are showed)
}

```

Figure 48. Example of a template with ILOG commands automatically generated as a C++ file

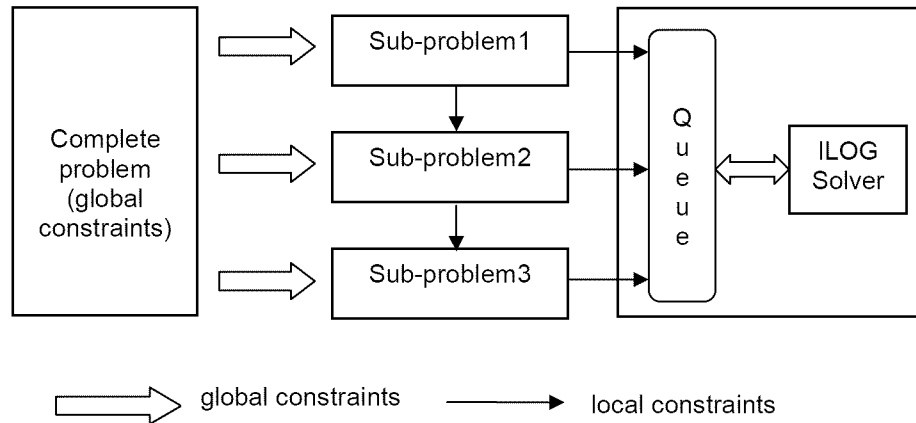


Figure 49. Design of problem decomposition by KF agent

One view of the iterative exchange of tentative solutions is as a distributed CSP (Durfee, 2001). The solution of one subproblem is a part of the task statement (a subset of local constraints) of another subproblem. Therefore related global constraints are passed to corresponding subproblems. Since (1) often the solution of one subproblem may affect solutions of others and (2) solving of subproblems is an iterative process therefore the solution of one subproblem is passed to all other subproblems as additional local constraints. These constraints may be new constraints or modified existing constraints in the task statement of an individual subproblem.

4.4. Translation Agent: Free Text Recognition

The translation agent delivers the user request to the KF agent in order to generate a solution set using ILOG Configurator. Since ILOG Configurator requires a *goal* presence, there is need in *explicit goal definition* defined by the user, e.g. minimize cost, time etc. (concrete goal formulation depends on the task and application domain).

4.4.1. Goal Description

Services for goal identification (required by ILOG and KF Agent see section 4.3) are provided by translation agent. It parses the user request and decomposes it into the parametric and structural request constituents. The user request parsing is performed by mapping the request terms into WordNet thesaurus WordNet, 2003) terms. User terms found in the thesaurus are supposed to correspond to ontology elements (O, Q, D, C); these terms are referred to the structural constituent. User terms not included in the structural constituent are considered as the parametric request constituent. This constituent is equal to user constraints on the ontology elements. As a rule, the parametric constituent is made up of user constraints on attribute domains or on attribute values.

A goal describes traditional optimization problems concerning minimization and maximization tasks. In the user request these tasks usually are represented by terms having the same meaning as minimum and maximum. Translation agent recognizes such the terms and terms being objectives of the optimization and formulates a goal. If the user request does not concern any optimization problems ILOG generates a feasible solution.

Figure 50 presents additional parameters related to the optimization process. There are three choices of optimization: minimize value of attribute, maximize, and no optimization. Optimized attribute can be constrained, i.e. attribute *Time* has to be minimized where minimum value is constrained (more than 7) in Figure 50.

```
<?xml version="1.0" standalone="yes"?>
<!--KSNet SPIIRAS-->
<Goal>

    <!-- Information about parameters to be optimized -->

    <Time>
        <MinOrMax>min</MinOrMax> <!-- Type of optimization: minimization-->
        <ConstraintValue>7</ConstraintValue> <!-- Limit: 7 -->
    </Time>
    <Cost>
        <MinOrMax>void</MinOrMax> <!-- No optimization -->
        <ConstraintValue>0</ConstraintValue>
    </Cost>
</Goal>
```

Figure 50. An example of XML file with additional parameters related to the optimization process

4.4.2. User Request Parsing

The translation agent uses the following sequence to parse a user request in English:

1. gets the user request as an input parameter;
2. extracts goals from the request;
3. extracts constraints from the request;
4. initializes WordNet;

5. extracts words corresponding to the types "Class" and "Task";
6. finds synonyms for every word;
7. saves extracted keywords to a file.

Request recognition includes (Figure 51):

- Omitting not significant words using stop-words list (Figure 52).
- Spelling and refinement (Figure 53). This module uses WordNet (WordNet, 2003).
- Stemming - definition of the main forms of the used words (Figure 54). It is based on the M.F.Porter stemming algorithm for suffix stripping (English). It was implemented as ActiveX component in C++.
- Usage of regular expressions enables attributes values recognition (Figure 55). The system will find values of characteristics (attributes) in the user request. The result is a list of characteristics, values and measures. E.g., <“accuracy”, “0.5”, “mm”>.
- Looking for terms in the ontology thesaurus taking into account synonyms (Figure 57).
- Structuring ("understanding" the user's intentions).
- Normalization of recognized attributes values -conversions of the user's measuring units into the system's ones (Figure 58).

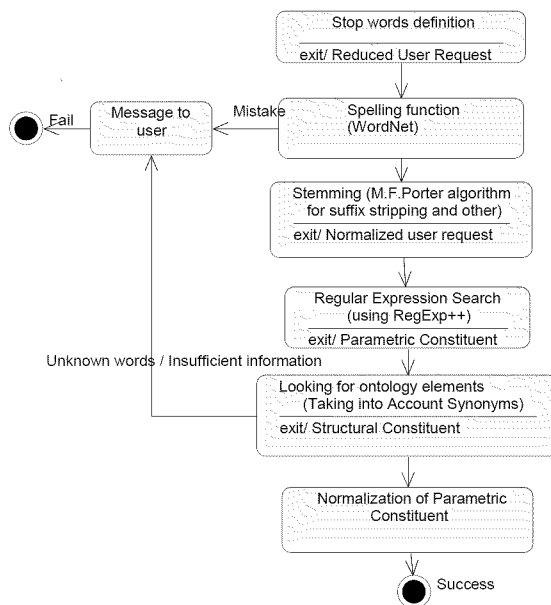


Figure 51. Sequence diagram of user request recognition

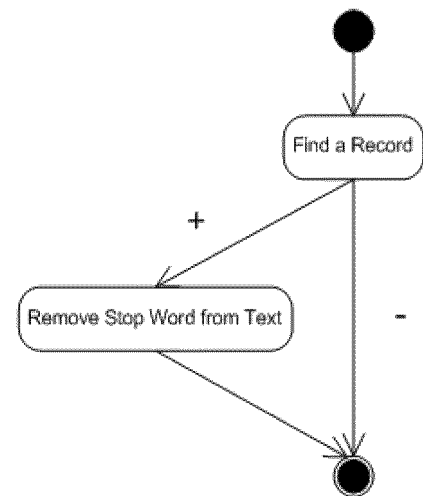


Figure 52. Sequence diagram of stop words processing

If the request is not recognized the procedure of interactive spelling is performed. The system informs the user about unknown words and proposes a list of suggestions.

To verify the system results and debug the program a list of XML files presenting results of user request processing is proposed. XML document consists of several blocks. Each block corresponds to some action and contains results of this activity.

A structure of the XML document is presented in (Figure 56):

Tags correspondence of the user request recognition procedures to the results is given below:

1. User request (tags <UserRequest> </UserRequest>)
2. Stop Words (tags <StopWords> </StopWords>)
3. Spelling check (tags <Spelling> </Spelling>)

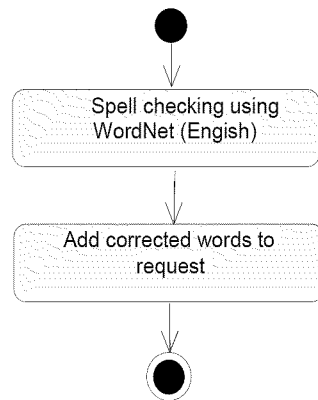


Figure 53. Sequence diagram of spelling check

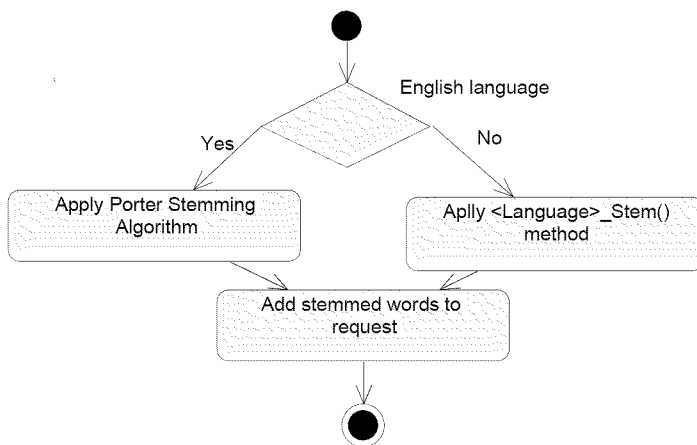


Figure 54. Sequence diagram of the stemming procedure

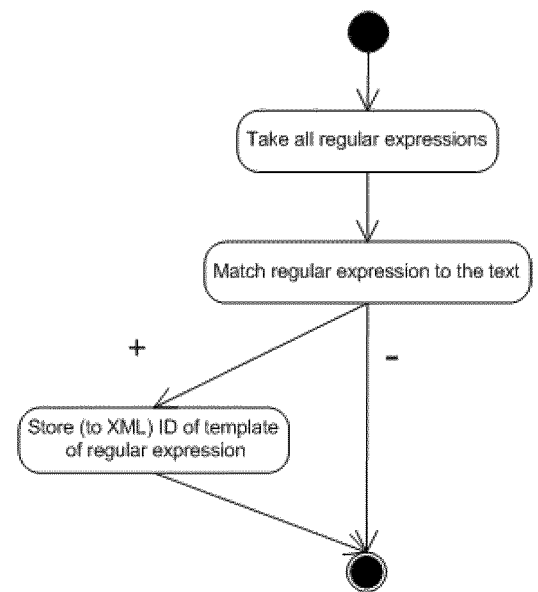


Figure 55. Sequence diagram of regular expressions usage

4. Stemming (tags <Stemming> </Stemming>)
5. Regular expressions (tags <RegularExpressions> </RegularExpressions>)
6. Structural and parametric constituents (tags <Ontology> </Ontology>)
7. Normalization (tags <Normalization> </Normalization>)

An example of XML file corresponding to the user request “I need a hospital located in 20 miles from the inhabited locality and no more than 50 km from water supply source” is presented in Figure 59: To present results of spelling procedure several errors (misspellings) were introduced in the request: “I need a hostpital located in 20 miles from the inhabitid locality and no more than 50 km from water supply source”

There are the following types of interactions with the user in the prototype:

1. Proposition to repeat the request and check of found spelling errors. The user will be returned to the request input form, request will be putted into the input field and text with the found error will be presented.
2. If the system can not recognize all the words of the request it can propose to the user to change the request or to ignore unrecognized words. E.g., “Unknown words: ... Please change your request – remove or change these words or press “Search” button to ignore them”.

```

<!-- User Request Processing -->
<Root>

<UserRequest></UserRequest>

<StopWords>
  <Word></Word>
  <Word></Word>
  ...
</StopWords>

<Spelling> <!-- only changed words-->
  <Word><In></In><Out></Out></Word>
  <Word><In></In><Out></Out></Word>
  ...
</Spelling>

<Stemming> <!-- only changed words-->
  <Word><In></In><Out></Out></Word>
  <Word><In></In><Out></Out></Word>
  ...
</Stemming>

<RegularExpressions>
  <Case>
    <ID>ID of used template of regular expression from table</ID>
    <String>Part of user request matched to this regular expression</String>
  </Case>

  <Case>
    <ID> </ID>
    <String> </String>
  </Case>
  ...
</RegularExpressions>

<Ontology>
  <Attribute><Name></Name><Value></Value></Attribute>
  <Class></Class>
  ...
</Ontology>

<Normalization>
  <String><In></In><Out></Out></String>
  <String><In></In><Out></Out></String>
  ...
</Normalization>

</Root>

```

Figure 56. XML document of user request recognition procedure

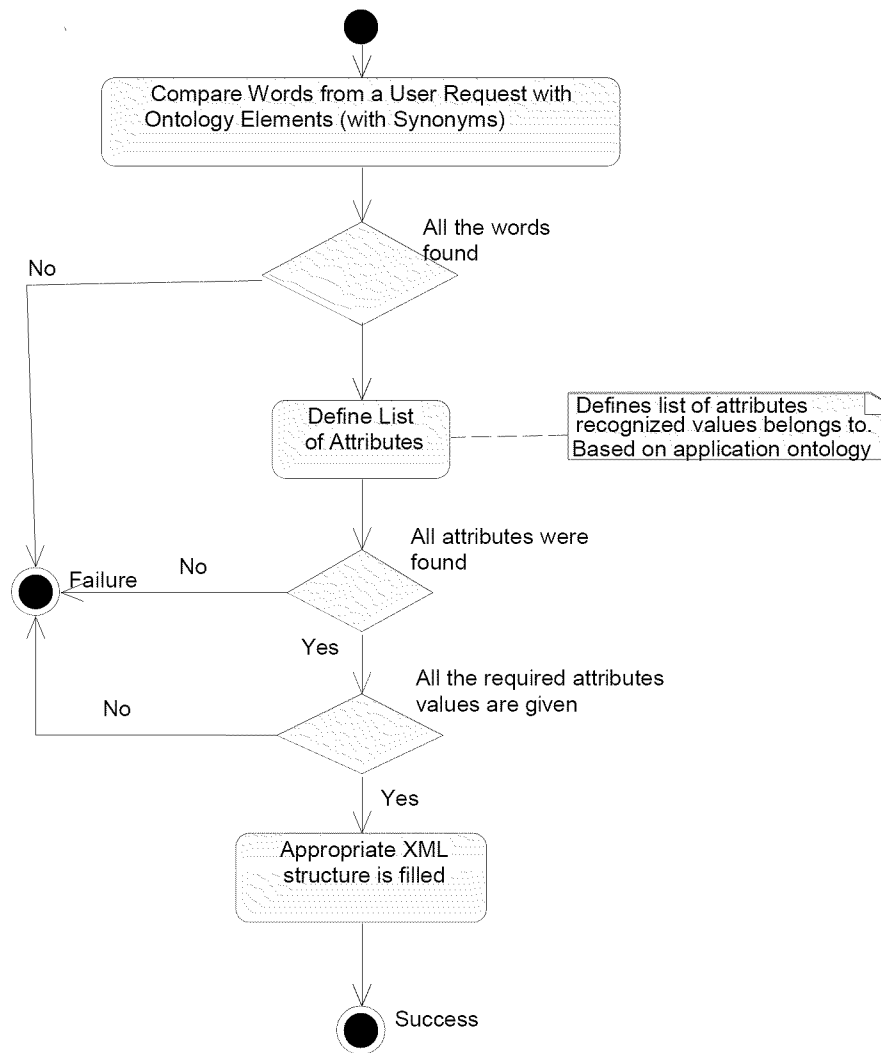


Figure 57. Sequence diagram of user request constituent formation

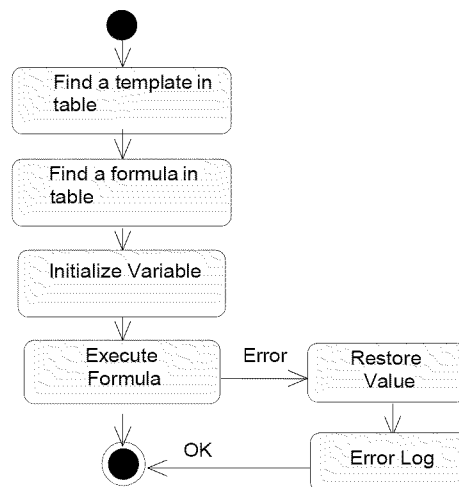


Figure 58. Sequence diagram of normalization procedure

```

<!-- User Request Processing -->
<Root>

<UserRequest> I need a hospital located in 20 miles from the inhabited locality and no more
than 50 km from water supply source </UserRequest>

<StopWords>
  <Word>I</Word>
  <Word>a</Word>
  <Word>in</Word>
  <Word>from</Word>
  <Word>the</Word>
  <Word>and</Word>
  <Word>no</Word>
</StopWords>

<Spelling>
  <Word>
    <In>hospital</In>
    <Out>hospital</Out>          <!-- Comment: right answer -->
    <Out>hospitals</Out>
    <Out>hospitable</Out>
    <Out>hospitably</Out>
    <Out>hospital's</Out>
    <Out>hostile</Out>
    <Out>hotpot</Out>
  </Word>
  <Word>
    <In>inhabited</In>
    <Out>inhabited</Out>        <!-- Comment: right answer -->
    <Out>inhabit id</Out>
    <Out>inhabit-id</Out>
    <Out>inhibited</Out>
    <Out>inhabits</Out>
    <Out>inhabit</Out>
    <Out>inhabiting</Out>
    <Out>inhabiter</Out>
  </Word>
</Spelling>

<Stemming>
  <Word><In>hospital</In><Out>hostpit</Out></Word>
  <Word><In>located</In><Out>locat</Out></Word>
  <Word><In>miles</In><Out>mile</Out></Word>
  <Word><In>locality</In><Out>local</Out></Word>
  <Word><In>source</In><Out>sourc</Out></Word>
</Stemming>

<RegularExpressions>
  <Case>
    <ID>1</ID>
    <String>20 miles</String >
  </Case>
  <Case>
    <ID>293</ID>
    <String>50 km</String >
  </Case>
</RegularExpressions>

<Ontology>
  <Attribute><Name>CityLocation</Name><Value>0.01</Value></Attribute>
  <Class>Pick&Place</Class >
  ...
</Ontology>

<Normalization>
  <String><In>20 miles</In><Out>32.18688 km</Out></String>
  <String><In>50 km</In><Out>50 km</Out></String>
</Normalization>

</Root>

```

Figure 59. An XML file example

4.5. Expert Assistant Agent: Knowledge Visualization and Groupware Support for Direct Knowledge Entry

4.5.1. Knowledge Visualization by Virtual Reality

Prototyping of the interface part of the described here approach is done via integration of the VRML technology and developed earlier Web-DESO ontology management system (Smirnov et al., 2002) as a major component of the user agent. Figure 60 demonstrates a taxonomy of Mobile Hospital Application Ontology and VRML-based screens (taxonomy of hospital equipment).



Figure 60. Example of prototyped VRML-based screens and Web-DESO

Current level of VRML technology (Web3D, 2002) maturity supports most of the requirements to knowledge visualization interface. It increases efficiency of knowledge entry due to combination of modeled images with our natural 3D perception of the world. It may support navigation over and through the knowledge, not supported by most of Web-based hypermedia systems, which include a navigational tool for the structure of the documents rather than for the knowledge nodes (Salis and Masili, 2002). On the other hand there are a number of VRML-based technologies for collaborative browsing (e.g., Blaxxun, 2002; DeepMatrix, 2002; Broll, 1997).

The common architecture can be represented as depicted in (Figure 61). Expert communicates with *user agent* by browsing a virtual reality world and interacting with its elements. All the expert's actions are interpreted by an appropriate *expert assistant agent* and transferred to the *mediator agent*. The latter performs matching of experts' actions and integrates those detecting contradictions.

Feedback from the *mediator agent* allows the experts to see each other and communicate: each expert is presented in the world via his/her avatar.

4.5.2. Expert Collaboration Support

To provide a possibility for an expert to present assurance of his/her knowledge the expert assistant agent prepares verbal and numeric scales for the expert work. Verbal and numeric scales are widely used in the decision support systems and KBs.

- When domain experts enter new knowledge in the internal KB. As it was described in the previous reports, knowledge storage was denoted as $I^{IKB} = \{i_j^{IKB}\}_{j=1}^{N_I}$, $N_I \in N$, where i_j^{IKB} are instances of classes. For each i_j^{IKB} a list of attributes with associated known / partially known (domains) / unknown values and degree of expert confidence $\{q, v, \omega\}_{kj}^{IKB}$ are attached.
- When domain expert set up correspondence between unrecognized user request and ontologies from the OL. Using degree of expert confidence allows to define quality of the system's response.
- To estimate a degree of correspondence of an AO to some domains or tasks & methods classifier elements.

To get an expert group confidence the following aggregation function of expert's local confidence is used:

$\omega = \sqrt[n]{\prod_{e=1}^n \omega_e}$, where ω - group confidence, n - a number of experts in the expert team, and ω_e - an expert's local confidence.

4.6. Ontology Management Agent: Ontology Library Maintenance

To provide compatibility of the used in the system "KSNet" functional constraints with ILOG notations a set of functions of the ontology management agent was extended.

Ontology management agent requires information from KSs extracted by wrappers. Every wrapper is created by software developers since it has to envelop specific, concrete KS. Implemented wrappers generate output message in XML format. This message has within the XML a tag <Code> (Figure 62) containing C++/ILOG compatible code. Objects and their properties extracted from the KS are described in this code.

The following ILOG Configurator functions were used as a basis for functional constraints in the system "KSNet":

- *IloNumVar getCardOf(IloComponentType t)* The number of components that are connected to the invoking port and are specialized as the type t.
- *IloNumVar getMax(const char* name)* Maximum value of the numeric attribute name on the components connected to the invoking port.
- *IloNumVar getMaxCard(const char* name)* Maximum value of the cardinality variable of the port name on the components connected to the invoking port.
- *IloNumVar getMin(const char* name)* Minimum value of the numeric attribute name on the components connected to the invoking port.
- *IloNumVar getMinCard(const char* name)* Minimum value of the cardinality variable of the port name on the components connected to the invoking port.
- *IloNumVar getSum(const char* name)* The sum of the values of the integer attribute name on the components connected to the invoking port.
- *IloNumVar getSumCard(const char* name)* The total value of the cardinality variables of the port name on the components connected to the invoking port.

The constraints are entered as follows: the ontology engineer selects a function from the list of available functions and assigns values of which attributes serve as input or output parameters of the function. An *expert assistant agent* sends a message to the *ontology management agent* to validate the syntax and to prepare the internal representation. The *ontology management agent* parses the constraint, checks the function's name, attribute names and attribute domains. If the constraint is correct, it prepares the internal ILOG representation of

the constraint. If the constraint is incorrect, it returns an error message to the *expert assistant agent*. The process of creating functional constraints in the system “KSNet” is illustrated by an UML sequence diagram (Figure 63).

```
<?xml version="1.0" standalone="yes"?>
<!--KSNet SPIIRAS-->
<code>/* Operating_Table suppliers*/
IloComponent object_8_1 = request.makeInstance (type_8, "object_8_1");
object_8_1.getNumAttr("attr_2_10").setValue(3); // attr=SupplierCapacity
object_8_1.getNumAttr("attr_2_13").setValue(11); // attr=SupplierCost4Item
object_8_1.getNumAttr("attr_2_14").setValue(11); // attr=SupplierTime4Item
object_8_1.getNumAttr("attr_2_12").setValue(9); // attr=DeliveryCost4Item
object_8_1.getNumAttr("attr_2_11").setValue(9); // attr=DeliveryTime4Item
object_3_2.getPort("port_3_2").connect(object_8_1);

if (print_wrapper_objects && wrp_obj_cur < wrp_obj_max) {
    wrp_obj_ptr [wrp_obj_cur] = &object_8_1;
    strcpy (wrp_obj_name [wrp_obj_cur++], "Operating Table supplier 1");
}

IloComponent object_8_2 = request.makeInstance (type_8, "object_8_2");
object_8_2.getNumAttr("attr_2_10").setValue(1); // attr=SupplierCapacity
object_8_2.getNumAttr("attr_2_13").setValue(21); // attr=SupplierCost4Item
object_8_2.getNumAttr("attr_2_14").setValue(8); // attr=SupplierTime4Item
object_8_2.getNumAttr("attr_2_12").setValue(22); // attr=DeliveryCost4Item
object_8_2.getNumAttr("attr_2_11").setValue(8); // attr=DeliveryTime4Item
object_3_2.getPort("port_3_2").connect(object_8_2);

if (print_wrapper_objects && wrp_obj_cur < wrp_obj_max) {
    wrp_obj_ptr [wrp_obj_cur] = &object_8_2;
    strcpy (wrp_obj_name [wrp_obj_cur++], "Operating Table supplier 2");
}
}
```

Figure 62. Example of wrapper’s output message in XML format with C++/ILOG compatible code

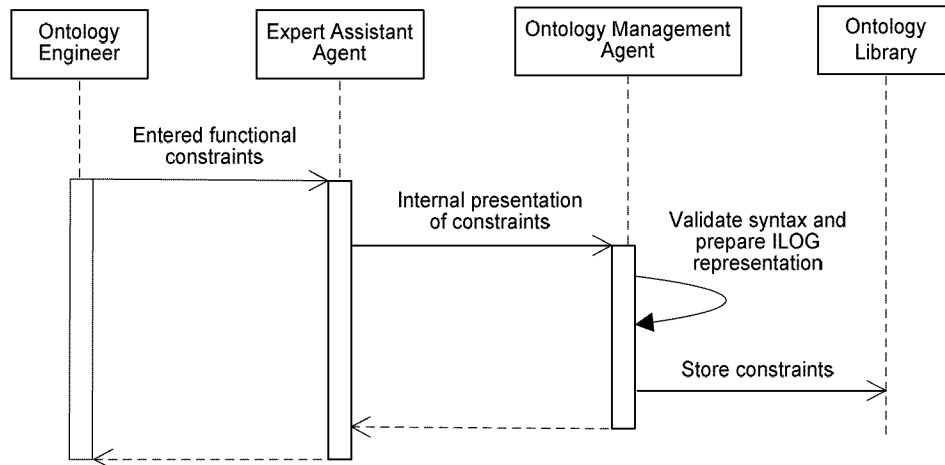


Figure 63. Scenario of functional constraints creation, conversion and storage

The KF agent extracts functional constraints (part of ontology) stored in OL in internal ILOG representation form. In Figure 64 an example of program code containing functional constraints generated by the KF agent is

presented. Comments of constraints for this code (lines started from “//”) have more human-readable form (Figure 64). Three presentation of one constraint (human readable, internal, and ILOG readable presentation) are presented in Table 10.

```
// Functional constraints

// comment: human readable format
// C++ code: machine (internal) format

// Supplier.add (SupplierTime == SupplierProg * SupplierTime4Item + DeliveryTime);
type_2.add (attr_2_9 == attr_2_7 * attr_2_14 + attr_2_5);

// Supplier.add (SupplierCost == SupplierProg * SupplierCost4Item + DeliveryCost);
type_2.add (attr_2_8 == attr_2_7 * attr_2_13 + attr_2_6);

// Supplier.add (SupplierProg <= SupplierCapacity);
type_2.add (attr_2_7 <= attr_2_10);

// Component.add (type_3.getNumAttr("GoodsCost") == port_Component_Supplier.getSum("SupplierCost"));
type_3.add (type_3.getNumAttr("attr_21_3") == port_3_2.getSum("attr_2_8"));

// Component.add (type_3.getNumAttr("GoodTime") == port_Component_Supplier.getMax("SupplierTime"));
type_3.add (type_3.getNumAttr("attr_21_15") == port_3_2.getMax("attr_2_9"));

// Hospital.add (type_1.getNumAttr("GoodsCost") == port_Hospital_Component.getSum("GoodsCost"));
type_1.add (type_1.getNumAttr("attr_21_3") == port_1_3.getSum("attr_21_3"));

// Hospital.add (type_1.getNumAttr("GoodTime") == port_Hospital_Component.getMax("GoodTime"));
type_1.add (type_1.getNumAttr("attr_21_15") == port_1_3.getMax("attr_21_15"));
```

Figure 64. Example of ILOG/C++ code generated by the KF agent on basis of the OL

Table 10. Different constraints presentation

Human readable presentation	Internal presentation	ILOG readable presentation
To configure a mobile hospital for certain quantity of patients with minimal quantity of beds and tables	(Patient.Quantity = Bed.Quantity AND Patient.Quantity/N<= Table.Quantity) OR (Patient.Quantity /N = Table.Quantity AND Patient.Quantity <= Bed.Quantity)	(port_1_17.getSum("attr_3_4") <= patients_per_1_table*port_1_18.getSum("attr_3_4") && attr_1_19==port_1_17.getSum("attr_3_4")) (port_1_18.getSum("attr_3_4") > patients_per_1_table*port_1_18.getSum("attr_3_4")) && attr_1_19==patients_per_1_table*port_1_18.getSum("attr_3_4"))

Web-DESO is intended for the knowledge representation by means of the OL notation, what ensures the ILOG compatible knowledge representation, and for the support of the operations over ontologies. The tool is implemented as a Web-based application.

The list of the supported operations:

- *Browsing*. Browsing OL and ontologies it stores.
- *Creation*. Creation of a new ontology.

- *Finding*. Search for desired string (or part of it) in the names of ontology elements. The operation consists in searching for names of ontologies, classes, and attributes coinciding with or including the sought for string, and searching for the string in descriptions of ontology, classes, and attributes.
- *Slicing*. Including indicated branches of the ontology taxonomy into a new ontology.
- *Merging*. Merging two or more ontology slices (or ontologies) into a single ontology.
- *Pruning*. Removal of the indicated ontology elements.
- *Modifying*. Maintenance of changes in ontology elements.
- *Import*. Import of ontologies from source formats into the internal representation. So far import from DAML+OIL format has been implemented.
- *Visualization*. Display in 3D view of an ontology (or an indicated ontology part).
- *Versioning*. Storage of ontology versions.

5. EXPERIMENTS

5.1. Experiments on Fusion-based Knowledge Logistics Technology

The evaluation of the full technology cycle prototype was done in accordance with the (Table 11). The table contains the components and scenarios of the system "KSNet" to be verified and expected results of the experiments. (Figure 65) illustrates the main components of the system to be verified. These experiments are based on the Binni scenario described in (5.2).

Table 11. Experiments for evaluation of the full technology cycle prototype

Module to Be Verified	Experiment Description	Expected Experiment Results
Ontology Management Agent: Operations on Ontologies	Verification of abilities to perform operations on ontologies	AO creation (5.3.1)
Ontology Management Agent: Ontology representation formalism	Verification of compatibility of the internal formalism with existing ones	Import/export of ontologies represented in DAML format into/from the internal format of the system "KSNet" (5.3.2)
Agent Negotiation Protocol	Evaluation of effectiveness of the developed agent negotiation protocol	Comparison of the time of negotiation using the developed and conventional protocols (5.4)
Translation Agent	Evaluation of effectiveness of the developed distributed architecture of the translation agent	Estimation of time spent by the translation agent for processing requests of different complexity (5.5)
Configuration Agent	Evaluation of effectiveness of the developed distributed architecture of the knowledge fusion agent	Estimation of KSNet configuration time for different numbers of KSs and different complexities of AO (sec. 4.1)
Knowledge Fusion Agent	Evaluation of effectiveness of the developed distributed architecture of the knowledge fusion agent	Estimation of the time of finding efficient solutions for tasks of different complexity (5.5)
Major System Scenario: Web-service architecture	Evaluation of ability for the system to act as a Web-service	Implementation of a Web-service – based interface for the system (5.7.3)
Major System Scenario: User request processing for different user preferences	Evaluation of feasibility of the developed user request processing scenario	Estimation of influence of user preferences on the request processing results (5.7.4)
Major System Scenario: User request processing with uncertainties	Evaluation of feasibility of the developed mechanism of uncertainty processing	Estimation of influence of uncertainty factors on the request processing results (5.7.5)
Major System Scenario: Agents' loads	Evaluation of degree of agents' workload	Definition of the most loaded agents (5.8)

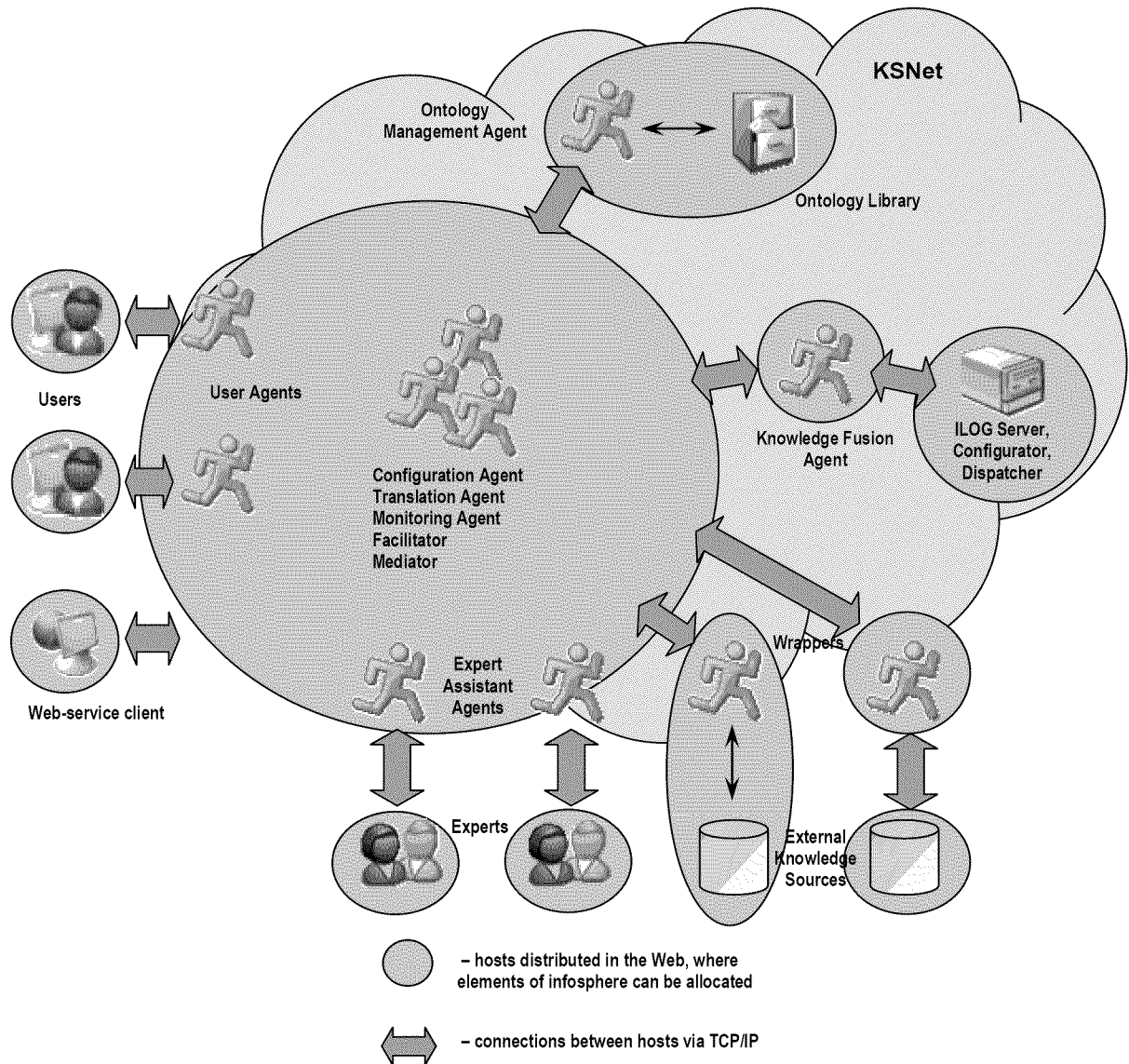


Figure 65. Main components of the system "KSNNet" to be verified

5.2. Binni Scenario as a Case Study

5.2.1. Health Service Logistics in Coalition Operations Other than War

Coalition operations other than war include a wide range of activities involving different people and organizations. In classical war operations the technology of control is strictly hierarchical, unlike operations other than war are very likely to be based on cooperation of a number of different, quasi-volunteered, vaguely organized groups of people, non-government organizations, institutions providing humanitarian aid but also army troops and official governmental initiatives. Here many participants will be ready to share information with some well specified community (Pechoucek, Marik, and Barta, 2001).

All joint doctrine and tactics, techniques, and procedures are organized into a comprehensive hierarchy as shown in (Figure 66). As a case study for the project the area of logistics was chosen because logistics problems can be widely used in a number of areas (e.g., other than war operations, supply chain management, transportation

systems, etc.) and it is of a high importance for completion of joint missions (JP 4-02.1, 1997). Focused logistics operations and/or Web-enhanced logistics operations address sustainment, transportation and end-to-end rapid supply to the final destination. Here the distributed information management and real-time information fusion to support continuous information integration of all participants of the operations are needed (DARPA, 2001).

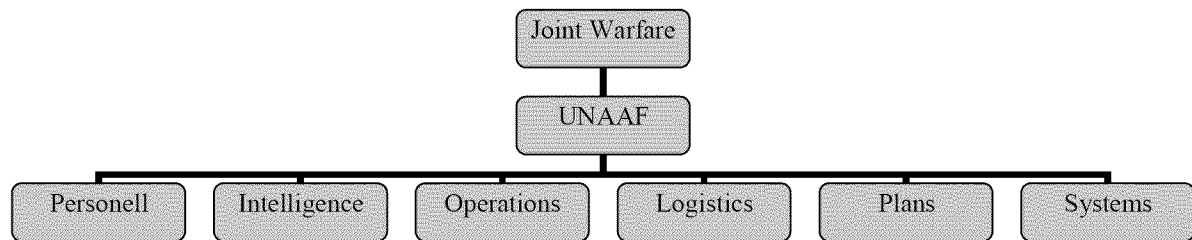


Figure 66. Joint doctrine hierarchy (adapted from JP 4-02.1, 1997)

Logistics support in coalition operations presents numerous challenges due to a variety of different policies, procedures and practices of the members of the operations, e.g., difference in doctrine, logistics mobility, resource limitations, differing stockage levels, interoperability concerns, competition between participants for common support.

In (JP 4-02.1, 1997) six major principles of joint activities logistics applying to operations other than war are selected. They include:

- *Objective.* There must be a clearly defined, decisive and attainable objective, and all the efforts of each operation member have to be integrated into the total effort of achieving strategic aims and cumulating in the desired end state.
- *Unity of effort.* There must be a close coordination of all the operation members provided leading toward the main goal and every subgoal.
- *Legitimacy.* Legitimacy involves sustaining the people's willingness to accept the right of the operation' leader to make and carry out decisions so that their activities would complement, not detract from, the legitimate authority of the leaders.
- *Perseverance.* In coalition operations strategic goals may be accomplished by long-term involvement, plans, and programs. Short duration operations may occur, but these operations have to be viewed as to their impact on the long-term strategic goals.
- *Restrain.* Coalition other than war operations put constraints on potential actions that can be undertaken by the operation's members to achieve their goals.
- *Security.* Security is a very important issue in coalition operations, especially in those related to healthcare and involving military forces. The operation's leaders and members have to ensure that they include security measures.

Coalition other than war operations may have different missions. E.g., they can be related to disaster relief, noncombatant evacuation, humanitarian assistance, peace operations, and other. For the project a task of disaster relief operation form the area of health logistics has been chosen; particularly, it is devoted to mobile hospital configuration.

5.2.2. General Description of Scenario

The aim of Binni scenario is to provide a rich agent technology environment, focusing on new aspects of coalition problems and new technologies demonstrating the ability of coalition-oriented agent services function in an increasingly dynamic environment. This was a main reason of using this scenario as a case study for the system KSNet in regard to the KF technology. The goal of the presented here scenario is to demonstrate an application of the developed KSNet-approach to operations other than war (OOTW) related problems. The considered task is a portable hospital configuration for a given location in the Binni region.

The nature of coalition-based operations has spanned a broad range of missions from war through OOTW. The OOTW cover a range of missions where sides are not in direct conflict but are required to perform a “neutral third party” operation. This is usually the result of a situation that is beyond the capability of the individual sides to resolve because it is an internecine issue or is beyond their individual resources. The missions may be further subdivided into war avoidance and humanitarian aid missions. The war avoidance operations cover the spectrum of “policing” activities that are required to restore “peaceful normality” in hostile situations between two or more population elements in conflict. In these circumstances the allied forces must act as an independent arbiter or “referee”. Countering terrorism and international crime may also be considered to lie within such missions because they can also be a significantly destabilising influence and may require the co-operation of international agencies in order to limit their insidious effects.

With the above factors in mind, “Binni – Gateway to the Golden Bowl of Africa” is a hypothetical scenario based on the Sudanese Plain (Figure 67). The countries of Gao, Agadez and Binni are fictitious, as are the events, organisations and personalities that lead to the crisis requiring UN intervention. However, it provides a backdrop against which to develop a number of exercises typical of those anticipated for future coalition force operations. Details of the scenario are given in a comprehensive document developed for the DARPA CoABS (Control of Agent Based Systems) program (Rathmell, 1999).

This case study was used in other works related to multiagent environments.

5.2.3. Binni Scenario for the System KSNet

The experimentation with Binni scenario is intended for demonstration of how the developed KSNet-approach can be used for support of coalition-based OOTW.

The following request is considered:

Define suppliers, transportation routes and schedules for building a hospital of given capacity at given location by given time.

An AO of this humanitarian task was built and connection of the found sources is performed. After this the request is processed.

An analysis of the built AO shows a necessity of finding and utilizing KSs containing the following information/knowledge:

- hospital related information (constraints on its structure, required quantities of components, required times of delivery);
- available UN and friendly suppliers (constraints on suppliers’ capabilities, capacities, locations);

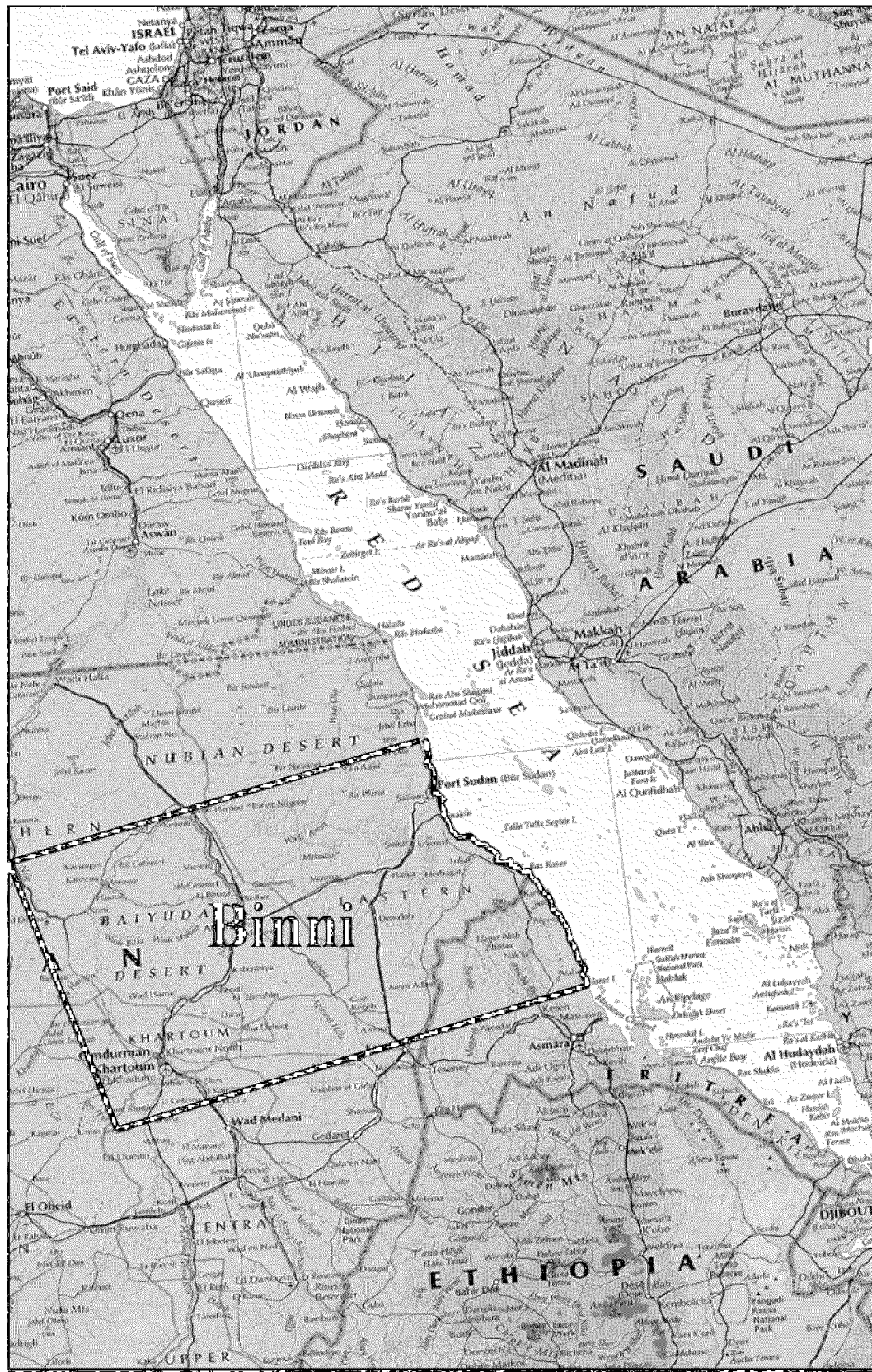


Figure 67. A map of Binni region

- available UN and friendly providers of transportation services (constraints on available types, routes, and time of delivery);
- geography and weather of the Binni region (constraints on types, routes, and time of delivery, e.g. by air, by trucks, by off-road vehicles);
- political situation, e.g. who occupies used for transportation territory, existence of military actions on the routes, etc. (additional constraints on routes of delivery).

Created AO can be later used to solve other tasks of the same nature in the Binni region. Example requests can be as follows:

- by what time a hospital/camp... of given capacity at given location can be built?
- where is better to build a hospital/camp...?
- find the best route to deliver something from point A to point B., etc.

As a result of the analysis of these problems the following modules were defined:

1. Portable hospital allocation.
This subproblem is devoted to finding the most appropriate location for a hospital to be built considering such factors as locations of the disaster, water resources, nearby cities and towns, communications facilities (e.g., locations of airports, roads, etc.) and decision maker's choice and priorities.
2. Routing problem.
This subproblem is devoted to finding the most efficient ways of delivery of the hospital's components from available suppliers considering such factors as communications facilities (e.g., locations of airports, roads, etc.), their conditions (e.g., good, damaged or destroyed roads), weather conditions (e.g., rains, storms, etc.) and decision maker's choice and priorities.
3. Hospital configuration.
This subproblem is very similar to that described in the previous report. It is devoted to finding the most efficient components for the hospital considering such factors as component suppliers, their capacities, prices, transportation time and costs and decision maker's choice and priorities.

Two scenarios of the problem solving are proposed (Figure 68). The first scenario is sequential and considers solving the subproblems one by one. The second scenario is iterative and assumes an iterative solving of the routing problem for suppliers used for building the hospital.

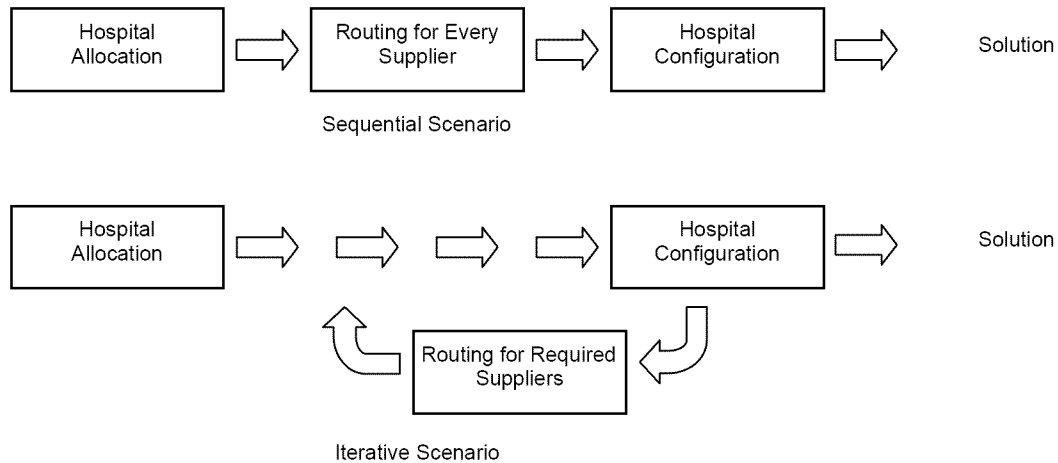


Figure 68. Portable hospital configuration scenarios

5.3. Experimentation on Ontologies

5.3.1. Application Ontology Creation

As it was mentioned before in the presented case study the hospital construction task is considered. In order to build AO for this case study, parts of ontologies corresponding to the described task were found in Internet's ontology libraries (Table 12). The analyzed ontologies represent a hospital in different manners. The resulting ontologies are shown in (Figure 69 - Figure 73). Operations on ontologies editing are shown in (Figure 74 - Figure 76).

Table 12. Ontologies used for building "hospital configuration" application ontology

Ontology	URL	Format
Clin-Act (Clinical Activity), the library of ontologies (Clin-Act, 2000)	http://saussure.irmkant.rm.cnr.it/onto/	KIF
Upper Cyc/HPKB IKB ontology with links to SENSUS, Version 1.4 (Cyc, 1998)	http://www.ksl-svc.stanford.edu:5915	Ontolingua (KIF)
Loom ontology browser, Information sciences Institute, The University of Southern California (Loom, 1997)	http://sevak.isi.edu:4676/loom/shuttle.html	Loom
North American Industry Classification System (NAICS) code, DAML Ontology Library (NAICS, 2001)	http://opencyc.sourceforge.net/daml/naics.daml	DAML
The UNSPSC Code (Universal Standard Products and Services Classification Code), DAML Ontology Library, Stanford University (UNSPSC, 2001)	http://www.ksl.stanford.edu/projects/-DAML/UNSPSC.daml	DAML
Web-Onto (Web-Onto, 2003)	http://eldora.open.ac.uk:3000/webonto	OCML

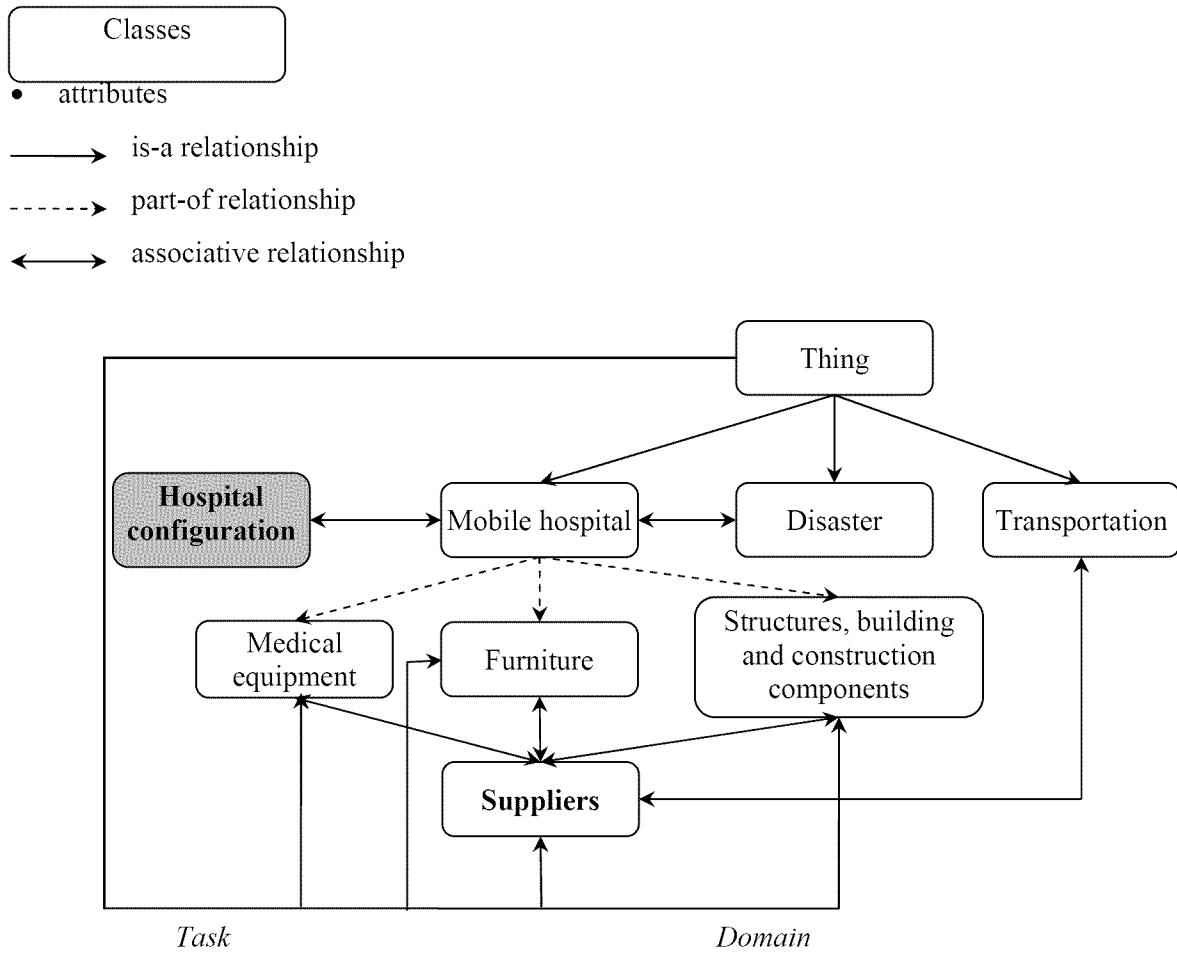


Figure 69. Application ontology: classes view

Class "Hospital Configuration" represents a task ontology expanded in (Figure 70).

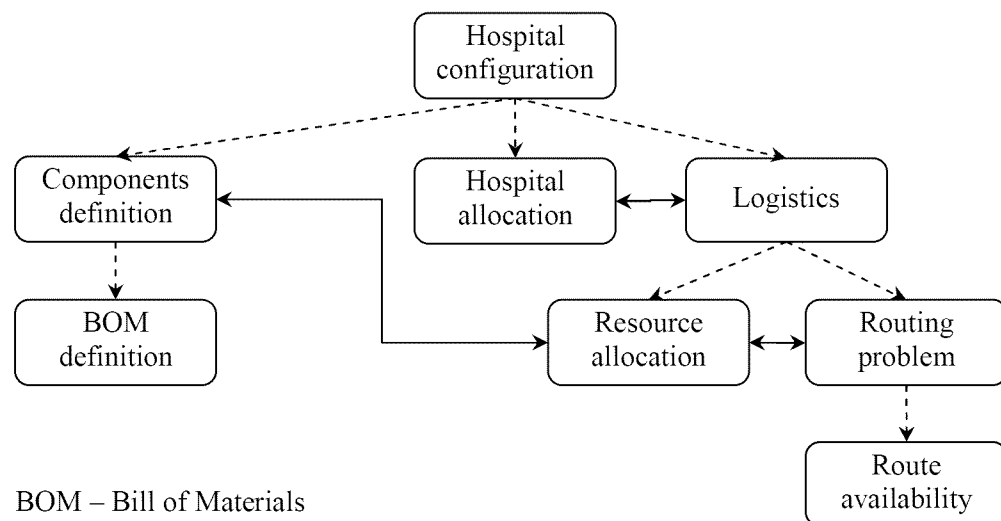


Figure 70. Task ontology "Hospital configuration"

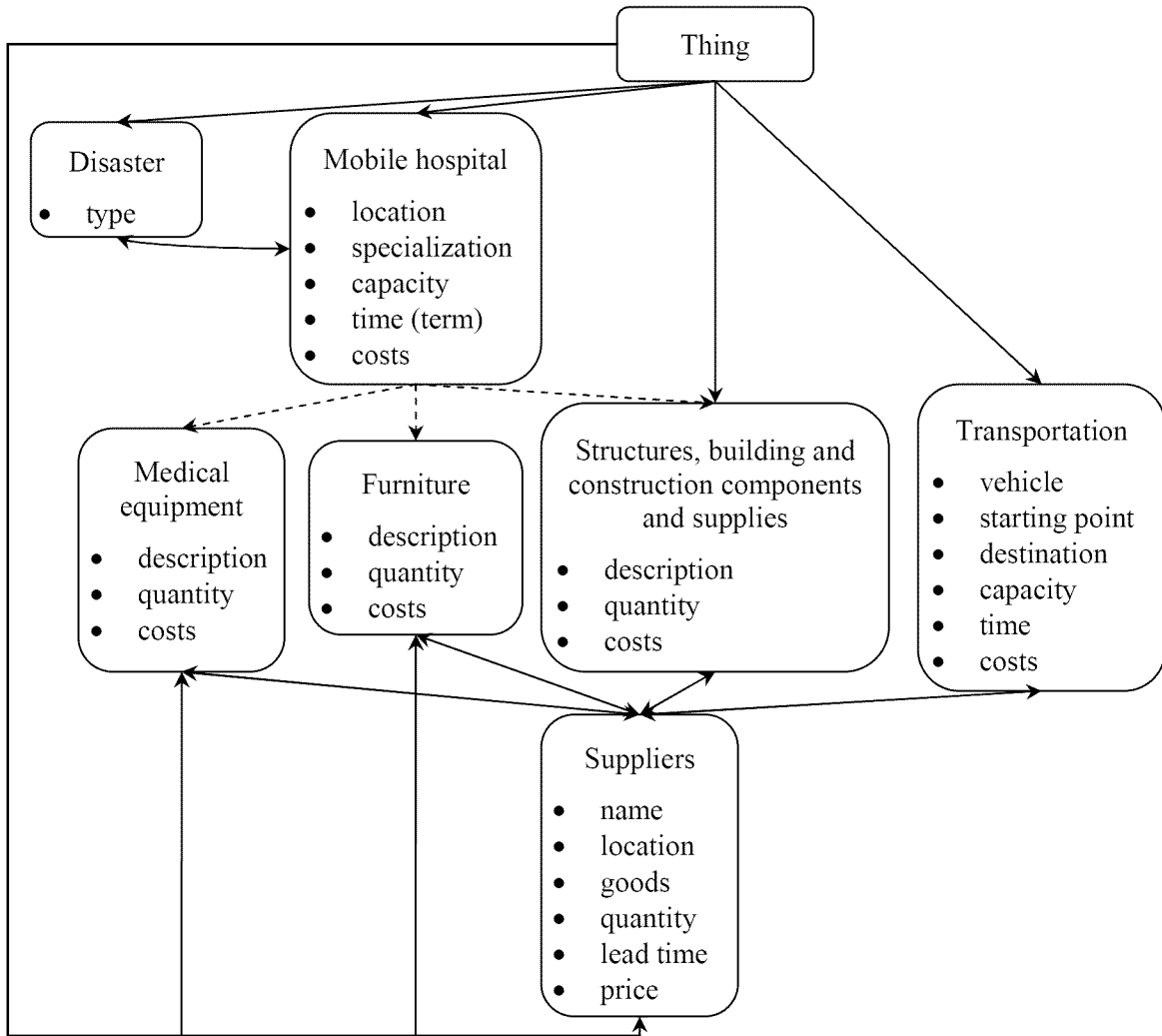


Figure 71. Domain ontology: classes and attributes view

The set of functions used to define application ontology attribute values during problem solving are presented in Figure 72. Functions from section “A” are used in the component definition subtask, functions from section “B” are used in the resource allocation subtask and functions from section “C” are used in the routing problem subtask. In this figure names of classes from AO are presented in square brackets, names of attributes are presented in curly brackets. “*f*” means name of function used for calculation of values attribute on the left (some constraints can use the same functions). “OR” means that set of functions is selected in dependence with given user constraints. \cup means that value of attribute depends on values of more then one attribute.

$$\begin{array}{ll}
\mathbf{A} & \text{Mobile hospital}. \{ \text{specialization} \} = f([\text{Disaster}]. \{ \text{type} \}) \\
\mathbf{B} & \left\{ \begin{array}{l}
[\text{Mobile hospital}]. \{ \text{costs} \} = f([\text{Medical equipment}]. \{ \text{costs} \}) \cup f([\text{Furniture}]. \{ \text{costs} \}) \cup \\
f([\text{Structures...}]. \{ \text{costs} \}) \\
[\text{Medical equipment}]. \{ \text{costs} \} = f([\text{Suppliers}]. \{ \text{costs} \}) \cup f([\text{Transportation}]. \{ \text{costs} \}) \\
[\text{Furniture}]. \{ \text{costs} \} = f([\text{Suppliers}]. \{ \text{costs} \}) \cup f([\text{Transportation}]. \{ \text{costs} \}) \\
[\text{Structures...}]. \{ \text{costs} \} = f([\text{Suppliers}]. \{ \text{costs} \}) \cup f([\text{Transportation}]. \{ \text{costs} \}) \\
\text{OR} \\
[\text{Mobile hospital}]. \{ \text{costs} \} = f([\text{Medical equipment}]. \{ \text{costs} \}) \cup f([\text{Furniture}]. \{ \text{costs} \}) \cup \\
f([\text{Structures...}]. \{ \text{costs} \}) \cup f([\text{Transportation}]. \{ \text{costs} \}) \\
[\text{Medical equipment}]. \{ \text{costs} \} = f([\text{Suppliers}]. \{ \text{costs} \}) \\
[\text{Furniture}]. \{ \text{costs} \} = f([\text{Suppliers}]. \{ \text{costs} \}) \\
[\text{Structures...}]. \{ \text{costs} \} = f([\text{Suppliers}]. \{ \text{costs} \})
\end{array} \right. \\
\mathbf{C} & \left\{ \begin{array}{l}
[\text{Transportation}]. \{ \text{costs} \} = f([\text{Suppliers}]. \{ \text{location} \}) \cup f([\text{Mobile hospital}]. \{ \text{location} \}) \\
[\text{Transportation}]. \{ \text{vehicle} \} = f([\text{Suppliers}]. \{ \text{location} \}) \cup f([\text{Mobile hospital}]. \{ \text{location} \}) \\
\cup f([\text{Suppliers}]. \{ \text{quantity} \}) \\
[\text{Transportation}]. \{ \text{time} \} = f([\text{Suppliers}]. \{ \text{lead time} \}) \cup f([\text{Suppliers}]. \{ \text{location} \}) \cup \\
f([\text{Mobile hospital}]. \{ \text{location} \})
\end{array} \right.
\end{array}$$

Figure 72.Set of functional constraints

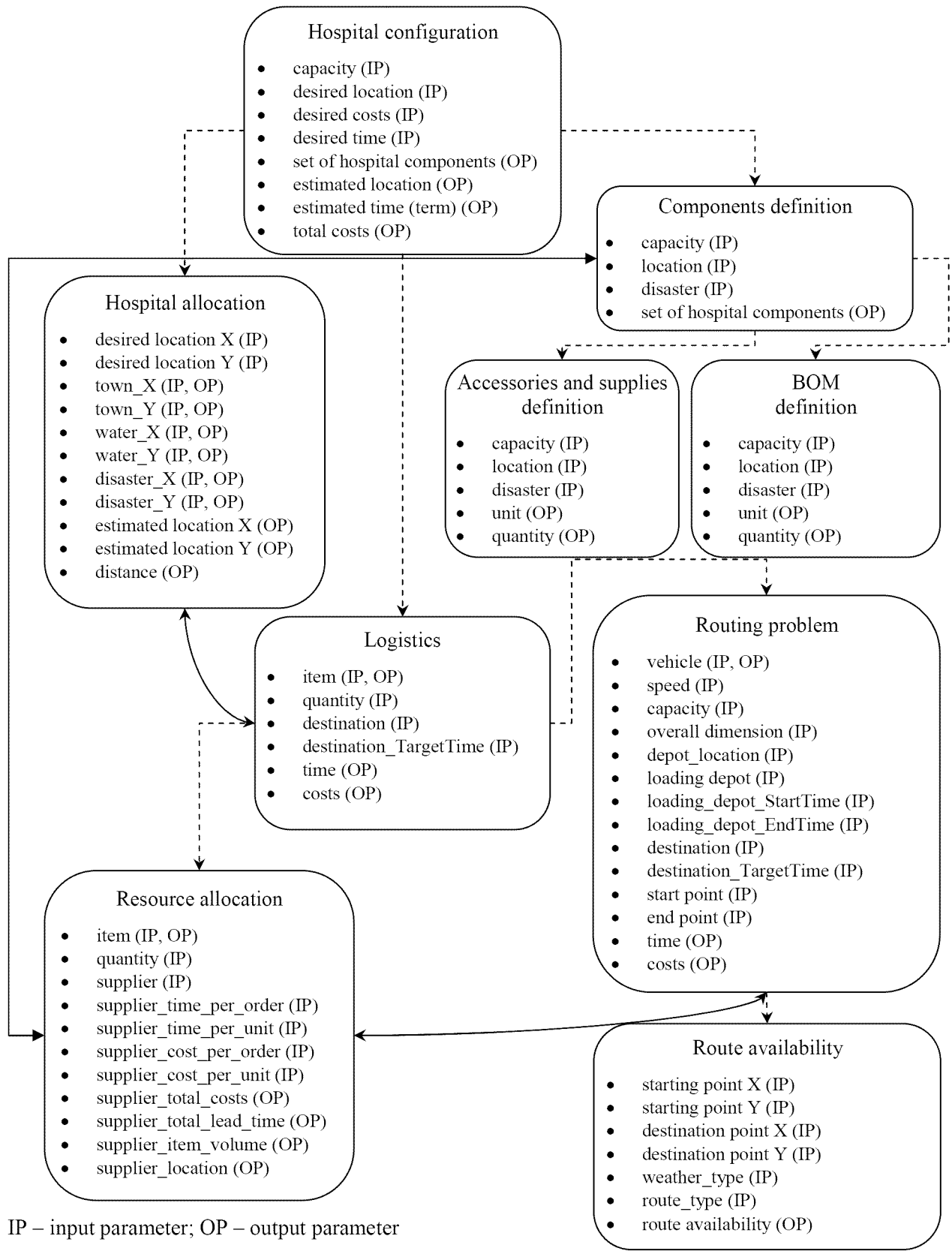


Figure 73. Task ontology: classes and attributes view

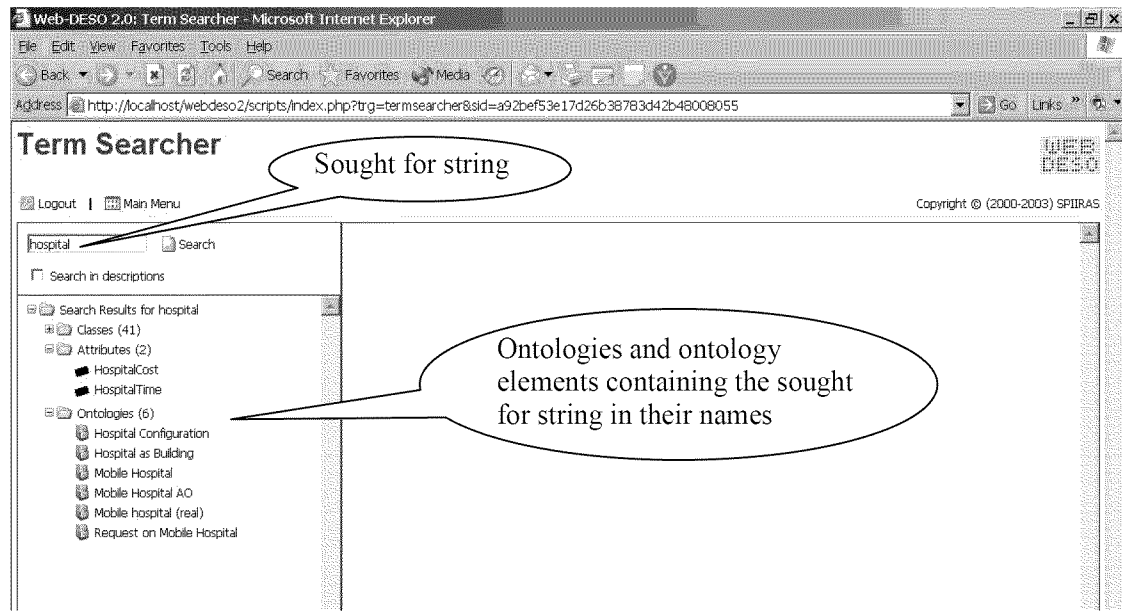


Figure 74. Finding operation

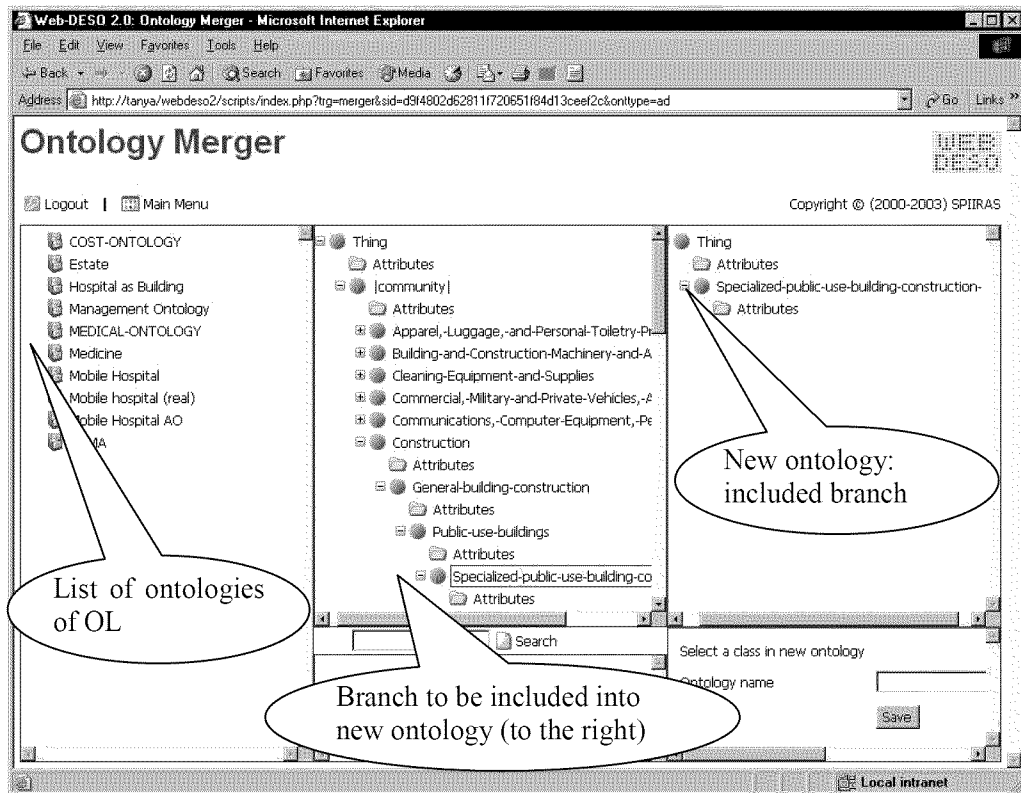


Figure 75. Merging operation: example for including a part of the existed ontology into a new ontology

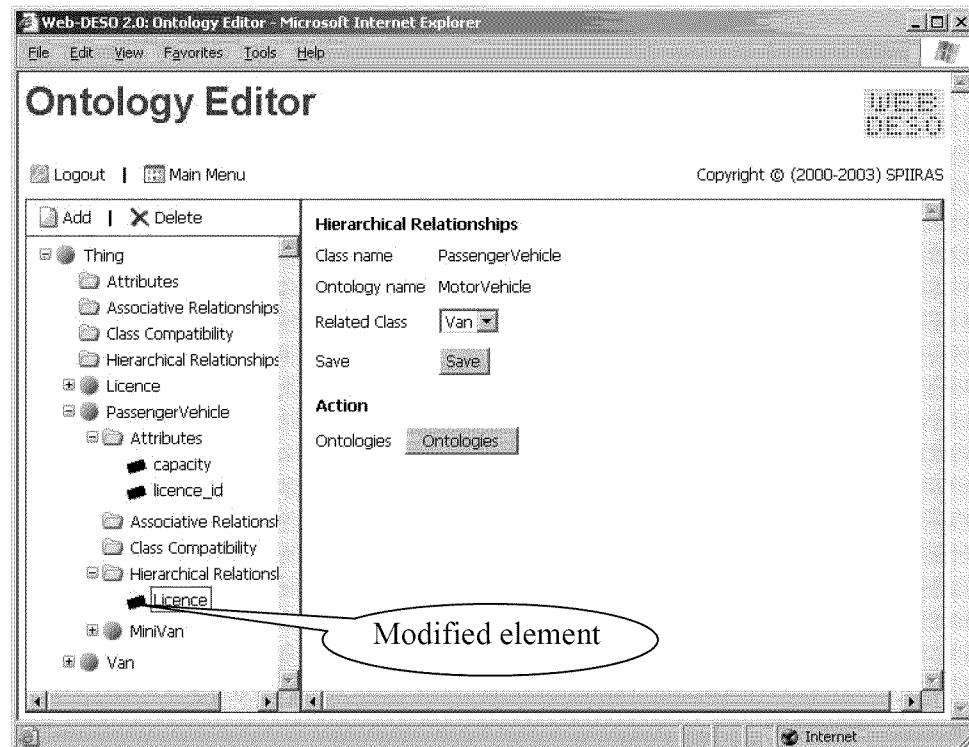


Figure 76. Modifying operation: example for editing hierarchical relationship

5.3.2. Verification of Compatibility of Internal Formalism with DAML

In order to verify compatibility of the developed ontology representation notation and existing notations such as DAML a number of experiments were performed. For this purpose a tool for importing ontologies represented in DAML format into the internal format of the Web-DESO tool (Interim Report # 4, 2002) was used. A number of ontologies required for the case study have been imported and two of them are shown here. The detailed description of the AO creation can be found in (Interim Report # 2, 2001).

The following translation peculiarities were taken into account during the import:

1. Some properties have a class name as a value of the attribute *range* and these classes do not have subclasses or attributes (for instance, properties *available-capacity*, *total-capacity*, *maximum-capacity* have *capacity-value* as attribute *range*). These classes are translated into domains with addition of appropriate constraints.
2. Some property have class names as attribute *range* and these classes have subclasses and/or attributes (for instance, properties *capacity-profile* has *capacity-value* as attribute *range*). These classes are added into the taxonomy as subclasses of *Thing* with addition of appropriate associative constraints.
3. Classes *position*, *position-profile* and *position-interval* are not translated since they are not needed in the chosen case-study.
4. Classes are translated into classes, with attributes *subClassOf* being used for the hierarchy and taxonomy creation.
5. *Properties* are translated into attributes, with appropriate constraints of attribute to class accessory being added.
6. During translation of some *properties* necessary domains and constraints of domain to attribute accessory are added or changed.
7. Functional constraints are added (for instance, calculation of the attribute *duration*).
8. After translation of the ontology 2 hierarchical relationships are added.

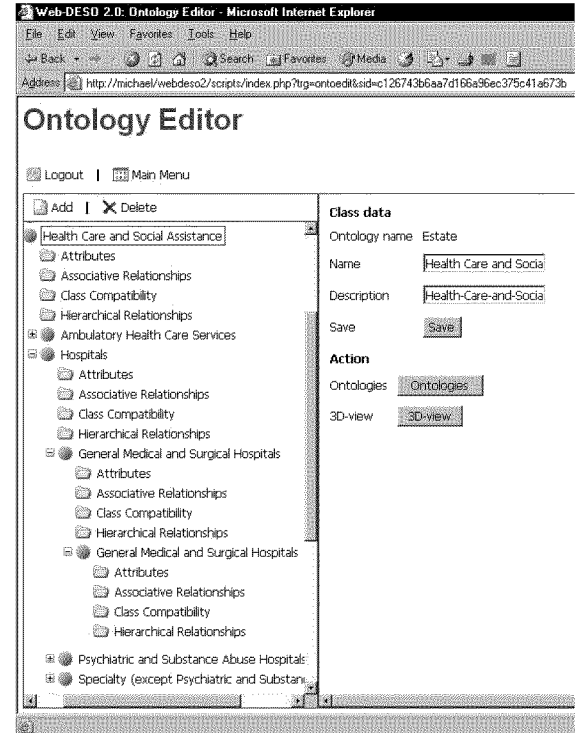
As a result "Health Care and Social Assistance" ontology represented in (Figure 77) was obtained and used for AO creation.

Summary of the possibility to convert knowledge elements from DAML into the internal format of the system "KSNet" is presented in (Table 13)

```
<?xml version='1.0' encoding='ISO-8859-1'?>
<!DOCTYPE rdf:RDF [
  <!ENTITY a 'http://www.daml.org/2001/03/daml+oil#'>
  <!ENTITY rdf 'http://www.w3.org/1999/02/22-rdf-syntax-ns#'>
  <!ENTITY b 'http://www.w3.org/2000/01/rdf-schema#'>
  <!ENTITY c 'http://opencyc.sourceforge.net/daml/naics#'>
]
<rdf:RDF xmlns:rdf="rdf;"
  xmlns:a="a;"
  xmlns:c="c;"
  xmlns:b="b;">

  <b:Class rdf:about="Health-Care-and-Social-Assistance;62"
    c:naics-code="62"
    b:label="Health Care and Social Assistance"/>
  </b:Class>
  <b:Class rdf:about="Ambulatory-Health-Care-Services;621"
    c:naics-code="621"
    b:label="Ambulatory Health Care Services">
    <b:subClassOf rdf:resource="Health-Care-and-Social-Assistance;62"/>
  </b:Class>
  <b:Class rdf:about="Hospitals;622"
    c:naics-code="622"
    b:label="Hospitals">
    <b:subClassOf rdf:resource="Health-Care-and-Social-Assistance;62"/>
  </b:Class>
  <b:Class rdf:about="General-Medical-and-Surgical-Hospitals;6221"
    c:naics-code="6221"
    b:label="General Medical and Surgical Hospitals">
    <b:subClassOf rdf:resource="Hospitals;622"/>
  </b:Class>
  <b:Class rdf:about="Psychiatric-and-Substance-Abuse-Hospitals;6222"
    c:naics-code="6222"
    b:label="Psychiatric and Substance Abuse Hospitals">
    <b:subClassOf rdf:resource="Hospitals;622"/>
  </b:Class>
  <b:Class rdf:about="Specialty-(except-Psychiatric-and-Substance-Abuse)-Hospitals;6223"
    c:naics-code="6223"
    b:label="Specialty (except Psychiatric and Substance Abuse) Hospitals">
    <b:subClassOf rdf:resource="Hospitals;622"/>
  </b:Class>
  <a:UniqueProperty rdf:about="naics-code"
    b:comment="North American Industry Classification System code"
    b:label="naics code">
    <b:domain rdf:resource="a:Thing"/>
    <b:range rdf:resource="b:Literal"/>
  </a:UniqueProperty>
</rdf:RDF>
```

DAML+OIL view



WebDESO view

Figure 77. Import of "Health Care and Social Assistance" ontology from DAML+OIL source format

Table 13. Possibility to convert knowledge elements from DAML into the internal format of the system "KSNet"

Element Groups	Elements form DAML+OIL
Elements supported by the notation accepted in the system KSNet	Class complementOf DatatypeRestriction Disjoint disjointWith maxCardinality maxCardinalityQ minCardinality minCardinalityQ ObjectRestriction Ontology Restriction versionInfo

Element Groups	Elements form DAML+OIL
Elements weakly supported by the notation accepted in the system KSNet	cardinality cardinalityQ DatatypeProperty domain range subClassOf unionOf
Elements not currently supported by the notation accepted in the system KSNet but such support is possible	hasClass hasClassQ hasValue Imports inverseOf ObjectProperty onProperty sameClassAs samePropertyAs toClass
Elements not currently supported by the notation accepted in the system KSNet and such support requires additional research	differentIndividualFrom disjointUnionOf equivalentTo intersectionOf oneof sameIndividualAs subPropertyOf TransitiveProperty UnambiguousProperty UniqueProperty

5.4. Experimentation on Agent Negotiation Protocol

To compare the results of the conventional CNP and constraint-based CNP the following example is considered. The experiments have been made using a multiagent development toolkit MAS DK (Gorodetski, et. al., 2001) which was chosen for a prototype of the system “KSNet”.

Configuration agent (CA) is supposed to obtain knowledge from three wrappers (W1, W2, and W3) with the time of knowledge acquisition being minimal. These criteria (time and costs) have been chosen because they are widely used. It is also preferable for the configuration agent to choose a cheaper deal among the deals with the same time:

CA: time \rightarrow min, costs \rightarrow min

The wrappers can make different offers such that the costs inversely depend on the time of knowledge delivery. This dependency is described by a table function given below:

W1: 30min/\$15

W2: 15min/\$20; 25min/\$10; 45min/\$5; ...

W3: 50min/\$25; 60min/\$15; 70min/\$10; ...

The resulting time and costs are calculated as follows:

`time = max(timeW1, timeW2, timeW3)`

`costs = sum(costsW1, costsW2, costsW3)`

The first scenario is performed in accordance with the conventional CNP (Figure 78). The configuration agent sends calls for proposals to all the wrappers concurrently. Besides description of the task to be performed each call contains additional constraints. In this case these constraints will contain the following:

`time → min`

The offers from wrappers will contain the following:

W1: 30min/\$15

W2: 15min/\$20

W3: 50min/\$25

The result will be 50 min and \$60.

The second scenario is similar to the first one but here the configuration agent does not send calls for proposals concurrently but consequently (Figure 79).

The first wrapper receives the following: `time → min` and replies with 30min/\$15.

The configuration agent analyses this offer and sends to the second wrapper the following request: `costs → min AND time ≤ 30`. Here the objective has become a constraint and a new objective is added. The wrapper replies with 25min/\$10.

The configuration agent analyses the offers it received and sends to the third wrapper the same request: `time ≤ 30 AND costs → min`. The wrapper replies with 50min/\$25. Here the wrapper cannot meet the requirements and it returns the best possible proposal.

The result is 50 min and \$50:

W1: 30min/\$15

W2: 25min/\$10

W3: 50min/\$25

The third scenario is performed in accordance with the presented in this section modified CNP. It contains concurrent conformation and iterative negotiation (Figure 81). At the first iteration the configuration agent sends calls for proposals to all the wrappers concurrently as in the first scenario, and the offers from the wrappers are the same:

time \rightarrow min

W1: 30min/\$15

W2: 15min/\$20

W3: 50min/\$25

After this the configuration agent analyses the results and sends new calls to the wrappers 1 and 2:

time ≤ 50 AND costs \rightarrow min

The wrappers reply as follows:

W1: 30min/\$15

W2: 45min/\$5

The result is 50 min and \$45:

W1: 30min/\$15

W2: 45min/\$5

W3: 50min/\$25

As it can be seen, proposed here constraint-based CNP allows achieving better solutions than the conventional CNP. In other words, given U_C , U_M – solution's utility for conventional and modified constraint-based CNP respectively, $U_M \geq U_C$ holds. On the other hand, it is obviously, that negotiation time for the proposed protocol increases. It can be seen that, given $T_{i=1..n}$ – response time of contractors (n – is the number of participating contractors), T_{man} – manager's response time, and T_C , T_M – negotiation time for conventional and modified constraint-based CNP respectively, $T_C \leq T_M \leq T_C + T_{max} + T_{man}$ holds, where $T_{max} = \max_{i=1}^n T_i$. This difference does not directly depend on the number of participating agents but it depends on T_{man} that in turn may depend on the task complexity and thereby on the number of participating agents. This dependency should be estimated for any particular case of the protocol usage.

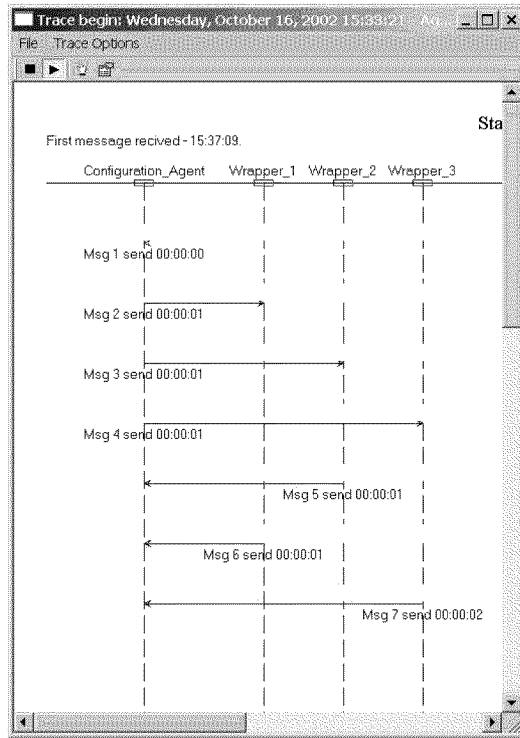


Figure 78. Experimentation with CNP: scenario 1

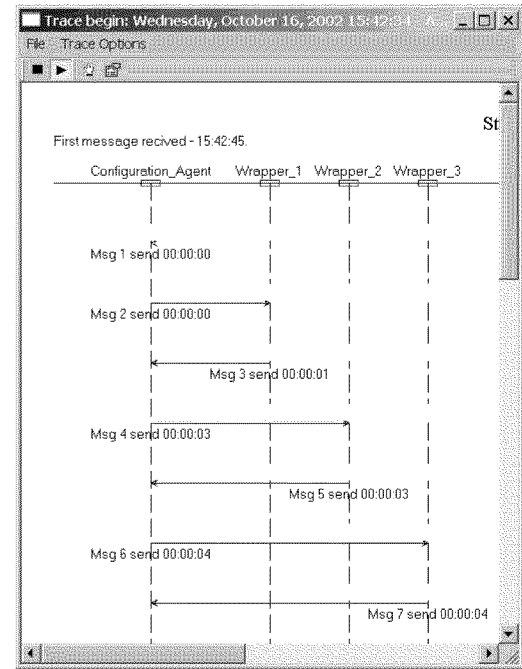


Figure 79. Experimentation with CNP: scenario 2

5.5. Experimentation on Translation Agent

The main goals of the translation agent during user request processing are related to recognition of the request. It is done via the following set of operations: Search for Regular Expressions, Stemming Procedure and Spelling Procedure. The experiments performed were motivated by these operations and presented in (Table 14).

Table 14. Experiments for evaluation of the request parsing by translation agent

Module to Be Verified	Experiment Description	Expected Experiment Results
Regular Expression	Search of substrings in the corpus of texts. Set of substring are defined by regular expression. Testing of work efficiency in different types of texts.	Estimation of search time. Search time has to be in reasonable limits for online text processing.
Stemming	Generate stems from the input texts. Defines time of work and number of stemmed words.	Estimation of search time. Search time has to be in reasonable limits for online text processing.
Spelling	Provide every erroneous word with	Estimation of search time. Search time has to be in reasonable limits for online text processing.

The set of request samples (corpus) was used for experiments with regular expressions, stemming, and spelling. The corpus has the following properties:

- corpus consists of 50 (49 in spell experiments) files in “plain text” format (.txt extension);
- the total corpus volume is 1492249 bytes (about 1.5 Mb);
- 33 files are of small size (1 KB) like major user requests.

5.5.1. Results of Experiments with Regular Expressions

This function performs a search and recognition for such constructions as "5 kilometers", "24 hours", etc. Several regular expressions were used in experiments:

1. $(\backslash d+)\backslash s^*m\backslash b$ This regular expression correspond the number $(\backslash d+)$, then spaces or tabs $\backslash s^*$, then letter "m", then boundary of word (space, comma, semicolon, etc.). E.g. this regular expression corresponds to such strings as "12 m", or "9 m", etc.
9. $(\backslash d+)\backslash s^*[a-zA-Z]^+$ This regular expression is used for searching number and first word after it.
10. $(\backslash w+)$ This regular expression is used for searching all words in text.

Result of using these 3 regular expressions for the entire corpus are presented in (Table 15). Time of work (second column) and number of retrieved substrings (three column) corresponds the results of treating the whole corpus.

Table 15. Results of experiments with regular expressions

Regular Expression	Time, sec	Substrings
$(\backslash d+)\backslash s^*m\backslash b$	0.046	4
$(\backslash d+)\backslash s^*[a-zA-Z]^+$	0.109	1577
$(\backslash w+)$	2.936	253750

It was concluded that time of search depends on the type of regular expression, though all three considered expressions were treated fast enough, since user request usually are much less than 1.5 MB. The time of the processing of user request formulated in (sec. 5.2.3) is less than 1×10^{-6} seconds.

5.5.2. Results of Experiments with Stemming

This module performs stemming of the words constituting the user request. Experiment shows that this operation is fast enough. File of the size about 1 Kb (like the request formulated in sec. 5.2.3) is stemmed in time less than 1×10^{-6} sec (Table 16). Total time of stemming of the whole corpus is 0.689 sec.

In order to demonstrate the effectiveness of this module some large texts were processed. They include several papers from the journal "Advances in Engineering Software" and two detective novels by Agatha Christie (Table 16).

Table 16. Results of experiments with stemming of large texts

Text	Time, seconds	Number of stemmed words	Bytes
A case-based procurement advisory system for construction	0.016	1702	33070
Design and development of system level software tool for DCS simulation	0.032	2856	52996
Negotiation within a multiagent system for the collaborative design of light industrial buildings	0.031	3088	60460
A simple algorithm for training fuzzy systems using input/output data	0.016	1427	36027
Modeling station duty officer operations assistant at Johnson Space Center	0.015	2656	50181

Text	Time, seconds	Number of stemmed words	Bytes
Utilization of a dependency-tracking language to reduce computational time during multidisciplinary design optimization	0.015	1844	35237
Visual and geometric reasoning for introductory figural space design	0.015	1546	42871
A dynamic e-Reporting system for contractor's performance appraisal	0.016	1737	37841
Supporting evolution in a multiagent cooperative design environment	0.016	2087	42315
The Mysterious Affair at Styles (Agatha Christie)	0.172	13778	347320
The Secret Adversary (Agatha Christie)	0.219	17543	452141

5.5.3. Results of Experiments with Spelling

Experiment showed that request processing by spelling module requires 0.1-1.0 seconds. Time of treatment depends on number of misspellings founded in text and number of spelling suggestions (Table 17).

Table 17. Results of experiments with Spelling

Request	Processing Time, seconds	Words	Misspellings	Suggestions
I need a hospital located in 20 miles from the inhabited locality and no more than 50 km from water supply source	0.125	22	2	15
Define supplierz, transportation routes and schedules for building a hospital of given capacity at given location by given time.	0.406	19	3	34

It can be concluded that the spelling is one the most time consuming operations (relatively to search of regular expression or stemming). So spelling can be used only for short text with 50-180 words, i.e. for user request which is usually short enough. The more valuable parameter for time of spelling's work is number of misspellings in the text. To cope with this problem it is required (1) add time constraint for spelling module or (2) add special method which can stop the spelling process from the outside.

5.5.4. Complex Experiments

The complex experiment was intended to check all modules of translation agent: (i) stop words removing, (ii) spelling, (iii) stemming, (iv) search of parameters via regular expressions, and (v) ontology term search. Eight user requests were used for this experiment. Example of user request (with spell errors) is "*I need a hospital located in 20 miles from the inhabited locality and no more than 50 km from water supply source*". The results of this experiment are presented in (Table 18). Time is given in the following format: "minutes:seconds,tenths of second". The most time consuming operations are spelling (with maximum time 4 and 7 seconds) and ontology search (with maximum work time 2 minutes 9 seconds).

It can be concluded that the most time consuming operation of translation agent is search terms in ontology.

Table 18. Complex Experiment with all modules of translation agent

N	StopWords	Spelling	Stemming	Regular Expression	Ontology Search
1	00:00.3	00:02.0	00:00.1	00:00.0	00:37.2
2	00:00.9	00:04.4	00:00.0	00:00.1	02:26.9
3	00:00.5	00:02.1	00:00.0	00:00.1	01:32.9
4	00:00.6	00:02.7	00:00.1	00:00.0	01:26.0
5	00:00.6	00:07.3	00:00.0	00:00.1	01:42.6
6	00:00.7	00:10.8	00:00.0	00:00.1	01:52.3
7	00:02.9	00:04.5	00:00.2	00:00.1	02:09.4
8	00:00.7	00:03.6	00:00.0	00:00.1	01:55.2

5.6. Experimentation on Knowledge Fusion Agent

The architecture of the KF agent is shown in (Figure 80). It utilizes such tools as ILOG Configurator, Dispatcher and Solver for constraint satisfaction and additional developed software modules for other tasks.

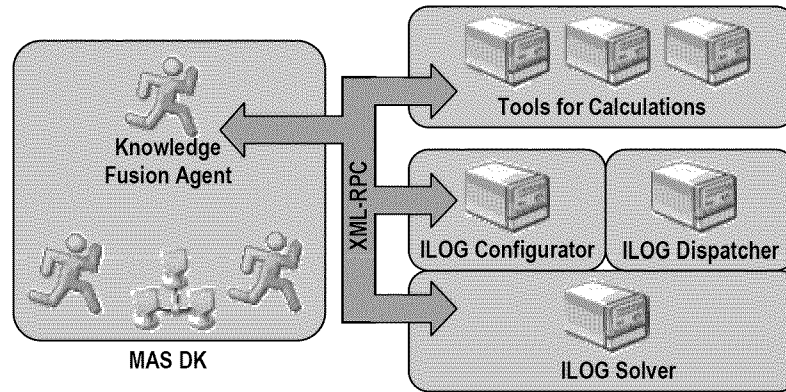


Figure 80. Architecture of the KF agent

The experimenting environment was the following:

- CPU: Pentium 4, 1.6 GHz;
- Operating memory: 1.0 GB;
- Hard disk drive: 50 GB.

The experiment was executed on the test ontology “Hospital” (Figure 82) as a part of case study based on Binni scenario (Interim Report # 4, 2002). This ontology contains all the ontology elements: classes, attributes, functional constraints, associative constraints, and compatibility constraints.

Figure 83 presents a diagram of the solver execution time (seconds) and the size of automatically generated file (kilobytes) against the number of objects extracted from KSs. When the number of objects (AO class instances) is more than 100 the ILOG execution time increases significantly. This is a usual commonplace when a task with enumeration of possibilities is solved by brute force. It can be concluded that heuristics and task restructuring are appropriate techniques. Well-grained task restructuring supposes that the set of solvable tasks instead of one heavy task is found. This problem division gives an opportunity to use distributed features of the system “KSNet” in full force.

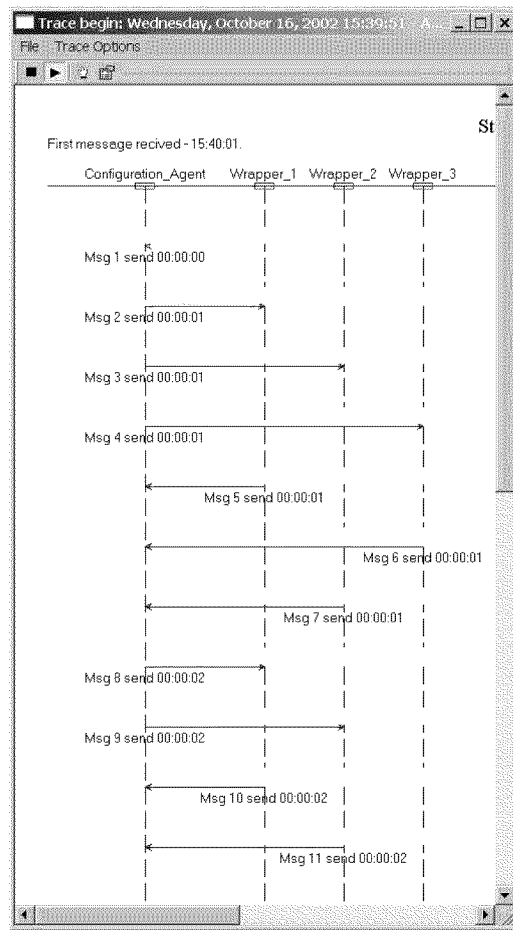


Figure 81. Experimentation with CNP: scenario 3

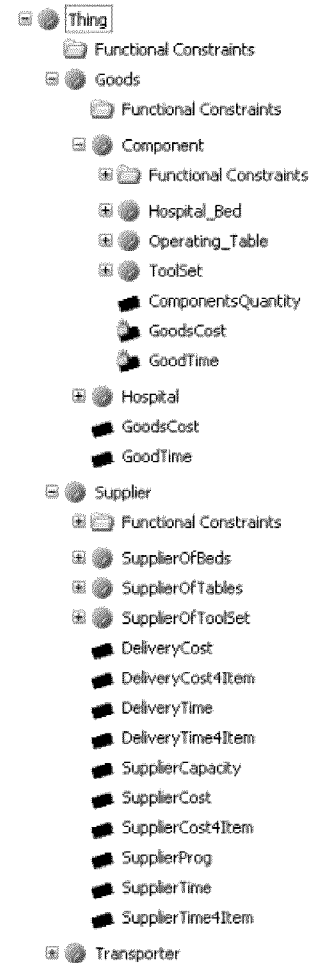


Figure 82. Taxonomy of the test ontology "Hospital" (classes, attributes, relations, etc.)

It was found that the compilation time does not depend on the number of objects essentially.

5.7. Experimentation with Complete Scenario of the System "KSNet"

5.7.1. Problem Structure and Decomposition

Detailed analysis of the case study based on the Binni scenario shows that complete problem includes a set of subtasks (Figure 70). Each of these subproblems is joined with other subproblems, so that the solution of one subproblem will be a basis for solving others. For example, the subproblem "Hospital configuration" depends on the "Hospital Allocation" and suppliers activity. In the same time suppliers activity depends on the available resources ("Resource allocation") and the cost of delivery ("Routing problem"). In turn, The solution of the subproblem "Hospital configuration" will allow making a more reliable basis in order to solve such problems as "Evacuation planning" or "Crisis management" (Figure 84).

To solve the case study subproblem "Hospital configuration" the KF agent gathers required data from distributed KSs with help of wrappers and uses a hospital ontology (Figure 82). Distributed KSs contain information about supplier products: operating tables, hospital beds, and tool sets (Figure 85). The supplier information includes the supplier production volume (capacity), delivery time, price per item, etc.

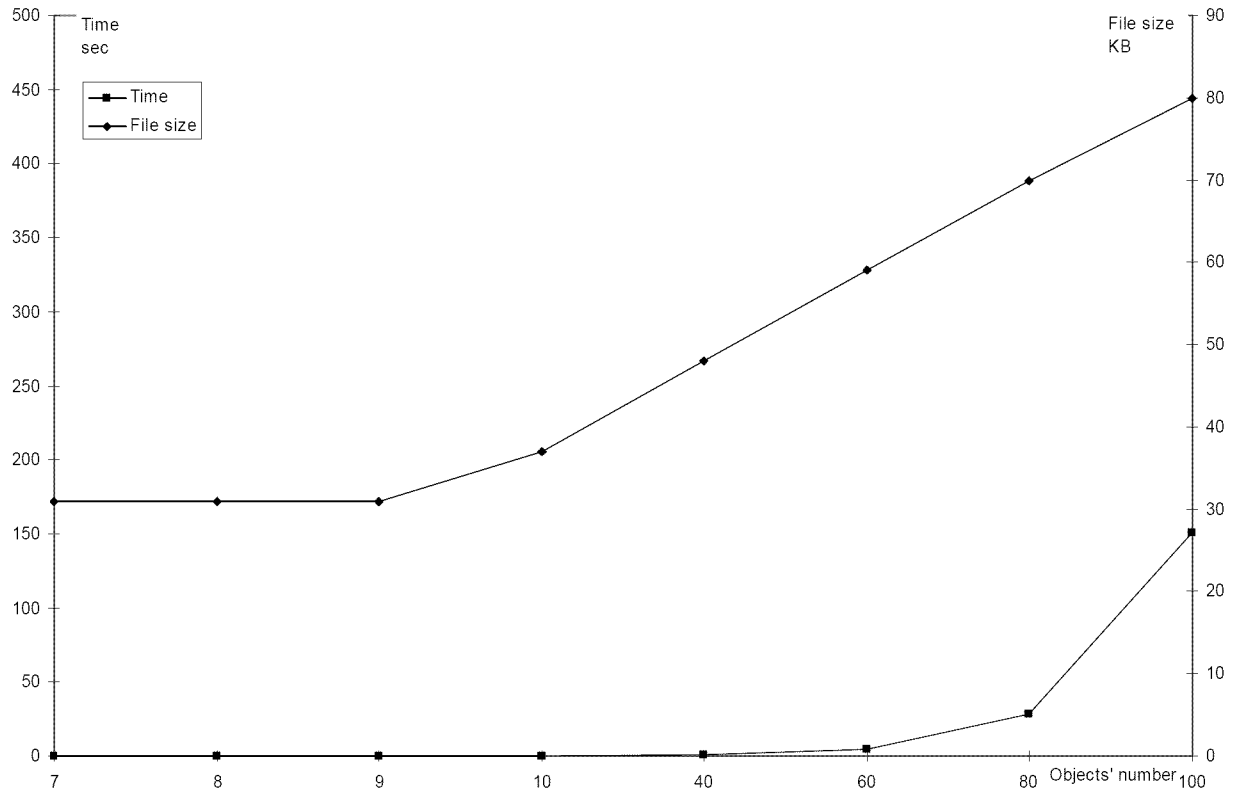


Figure 83. Plot of solver execution time (Time) and size of generated file (File size) against number of objects extracted from KSs

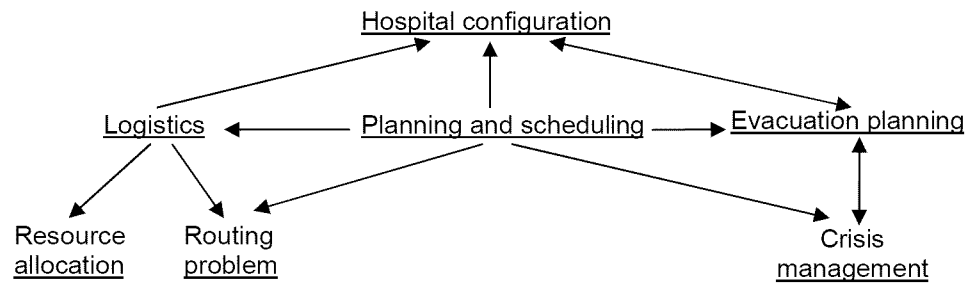


Figure 84. Example of solution's dependence of the "Hospital configuration" subproblem

The experiment shows (Figure 83) that solving the complete problem is beyond the computational capability of a modern computer. Since constraint satisfaction is NP-complete in general (Yokoo, 1999) therefore special approaches solving complete problem are needed: using heuristics, local search, and problem decomposition. In (Durfee, 2001) problems concerning the "task sharing" were enumerated. In the same time these problems are intrinsic properties of the problem decomposition:

- Problems will often require backtracking upward in the abstraction hierarchy, because the solution of one can affect the solution of others;
- Problems usually cannot be decomposed into equal-sized subproblems;
- The process of decomposing problems, distributing subproblems, and collecting results can require substantial time.

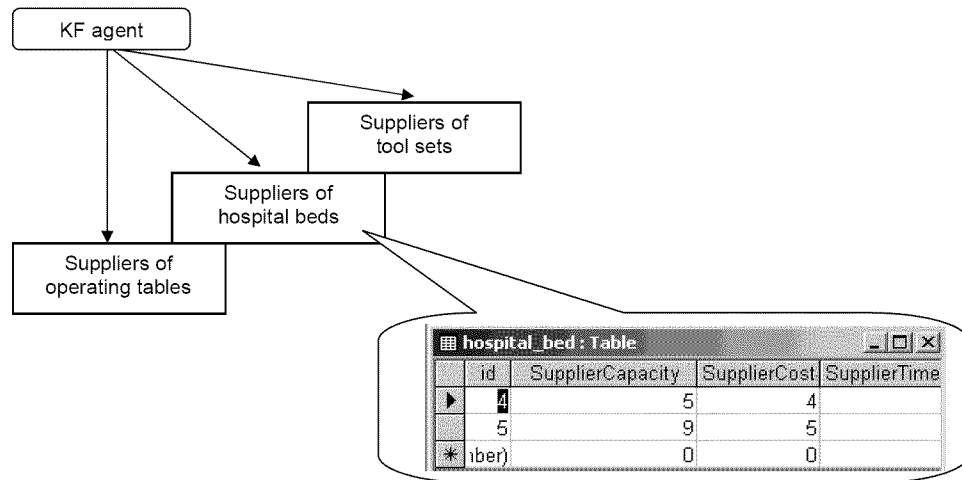


Figure 85. Distributed KSs used by the KF agent

It is proposed that the problem decomposition results in more efficient resolution, better solution in given CPU time.

5.7.1.1. Capacity Definition and BOM Definition

In the viewed example the following hospital Bill of Materials (BOM) structure (Figure 86) is considered as default:

1. Hospital bed (10 pieces per 10 places)
 - 1.1. Bed frame (1)
 - 1.2. Mattress (1)
11. Operating table (1 piece per 10 places)
 - 1.3. Table units (1)
 - 1.4. Table frame (1)
12. Medical toolset (5 pieces per 10 places)
 - 1.5. Plastic accessories
 - 1.6. Metal accessories

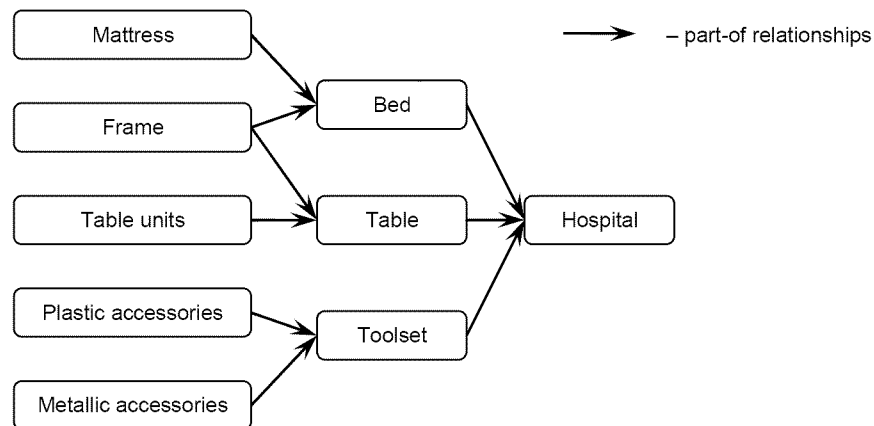


Figure 86. BOM structure

Depending on the disaster type the BOM structure may vary.

5.7.1.2. Hospital Allocation

This subproblem is devoted to finding the most appropriate location for a hospital to be built considering such factors as locations of the disaster, water resources, nearby cities and towns, communications facilities (e.g., locations of airports, roads, etc.) and decision maker's choice and priorities. This task is considered in the following way:

There are several points (defined by attributes: (1) position X, Y and (2) type $City_i$, $Disaster_j$, $Water_k$ (City, Disaster, and Water). There is a set of points suitable for mobile hospital location S_i .

Goal function (maximal distance):

$$MaxDist \{S_i\} = \max (\min (Dist (S_i, City_i)); \min (Dist (S_j, Disaster_j)); \min (Dist (S_k, Water_k)))$$

The goal is to find the best S_i minimizing $MaxDist$.

Minimize $MaxDist \{S_i\}$ selecting best hospital location S_i .

$$MaxDist = \max \left\{ \begin{array}{l} \min_j (Dist(S_i, City_j)) \\ \min_k (Dist(S_i, Disaster_k)) \\ \min_l (Dist(S_i, Water_l)) \end{array} \right\} \rightarrow \min$$

This problem is solved using ILOG Solver. Results of experimenting with different task complexities are shown in (Table 19).

Table 19. Statistics of routing problem being solved by ILOG Solver

Number of cities	MinDist (degree)	ILOG execution time (seconds)
2	1.78	0.1
10	1.78	0.1
40	1.78	0.1
100	1.78	0.4
150	1.69	0.6
200	1.69	1.1
300	1.69	3.1
400	1.67	7.2
500	1.67	12.8
600	1.56	18.5
700	1.56	36.1
742	1.56	55.9

5.7.1.3. Resource Allocation

This subproblem is devoted to finding the most efficient hospital configuration considering such factors as:

1. Hospital requirements (type and quantity of material or goods, ex. five operational tables or four tool sets);
2. Supplier properties (productivity (number of available goods), cost or unit price (cost per one goods, and cost per one order), travel cost (per one goods, and travel cost per one order)), availability of supplier.
3. Optimization parameter (cost or time)

The output (result) is

1. The set of pareto-optimal combination of suitable suppliers, where set of parameters (number of ordered goods, types of ordered goods, manufacturing and transport costs) have been defined for every supplier.
2. Total cost and time.

This task is solved using ILOG Configurator. The set of goals was formulated as follows:

1. configure the hospital: attach components (find most appropriate components)
2. define and configure component: attach suppliers (find most appropriate suppliers)
3. define supplier (define supplier's parameters)

5.7.1.4. Routing and Weather

This subproblem is devoted to finding a pareto-optimal set of routes of delivery of the hospital's components from chosen suppliers considering such factors as communications facilities (e.g., locations of airports, roads, etc.), their conditions (e.g., good, damaged or destroyed roads), weather conditions (e.g., rains, storms, etc.) and decision maker's choice and priorities.

This task was solved using ILOG Dispatcher. This tool is specially designed for solving transportation related tasks. ILOG Dispatcher offers a variety of solution search and solution generation heuristics (Table 20, Table 21).

Table 20. ILOG Dispatcher heuristics for generating solutions

Heuristics	Description
Enumeration heuristic	The enumeration heuristic builds a solution to the problem using an algorithm that completely explores the search space using backtracking. This method should only be used for small problems.
<i>Savings heuristic</i>	The savings heuristic considers the trade-off between more vehicles with shorter routes and fewer vehicles with longer routes.
Sweep heuristic	<p>The sweep heuristic builds routes by sweeping around the depot: Let O be a site from which vehicles leave (usually a depot), and let A (different from O) be another site which serves as a reference. Sort all the sites S in the routing plan by increasing angle AOS. Put the result in a list L. The visits corresponding to the sites in L will be allocated to the vehicles in that order as long as constraints are respected. If all vehicles have been used, the remaining visits are constrained to be unperformed. If one of these visits must be performed, the goal fails.</p>

Heuristics	Description
Nearest-to-depot heuristic	<p>The nearest-to-depot heuristic builds routes by adding the visits close to the depot first.</p> <p>For all vehicles:</p> <p>Denote the vehicle to be considered by w.</p> <p>Start with a partial route consisting of the departure from the depot.</p> <p>Find the visit v which is closest to the starting point of the current partial route of w. If it is not possible to find such a visit without violating constraints, close the current partial route of w, choose another empty vehicle and go to step 2. If no empty vehicles remain, the goal fails.</p> <p>Add v to the end of the partial route.</p> <p>Go to step 3.</p> <p>If all vehicles have been used, the remaining visits are constrained to be unperformed. If one of these visits must be performed, the goal fails.</p>
Nearest addition heuristic	<p>The nearest addition heuristic is very similar to the nearest-to-depot heuristic. It often gives better results because the visit added to the route is the one closer to the end of the route, not the one closer to the depot.</p> <p>For all vehicles:</p> <p>Denote the vehicle to be considered by w.</p> <p>Start with a partial route consisting of the departure from the depot.</p> <p>Find the visit v which is closest (that is, least costly to get to) to the end of the current partial route of w. If it is not possible to find such a visit without violating constraints, close the current partial route w, choose another empty vehicle and go to step 2. If no empty vehicles exist, the goal fails.</p> <p>Add v to the end of the partial route.</p> <p>Go to step 3.</p> <p>If all vehicles have been used, the remaining visits are constrained to be unperformed. If one of these visits must be performed, the goal fails.</p>
Insertion heuristic	<p>The insertion heuristic works by inserting each visit (in the order they were created) at the best possible place, in terms of cost:</p> <p>Let all vehicles have empty routes.</p> <p>Let L be the list of unassigned visits.</p> <p>Take a visit v in L.</p> <p>Insert v in a route at a feasible position where there will be the least increase in cost. If there is no feasible position, then the goal fails.</p> <p>Remove v from L.</p> <p>If L is not empty, go to 3.</p>

Table 21. ILOG Dispatcher heuristics for optimal solution search

Heuristics	Description
First accept	This method accepts any legal improving move and so is not too expensive in terms of computational cost. Thus, it is preferred when the problem is very large or an optimized solution is required as quickly as possible.
Best accept	This method accepts the most profitable (in terms of cost) legal improving move and so is more expensive in terms of computational cost than the first accept method just described. For some problems, however, the results can be better than

Heuristics	Description
	those provided by first accept. You should experiment on your own problem to find the best fit.
Tabu search	Tabu search heuristic allows degrading moves to be accepted by the local search process and attempt to avoid moving the search cycle to places it has previously been. To prevent cycling ILOG tabu search heuristic uses the popular technique of the tabu list. The tabu list is a list of "features" of a solution that are forbidden, or alternatively, that must be present. Features are added and dropped from a list when a neighborhood move is made.
<i>Guided local search</i>	Guided local search is an alternative to tabu search for allowing the search process to move out of local minima. As in tabu search, how the search can move around is restricted. Guided local search works by making a series of greedy searches, each to a local minimum. However, it reduces a different cost function from the original. If the original cost function is represented by c , then guided local search attempts to reduce the cost $c + wp$, where p is a penalty term that is adjusted every time a local minimum is reached, and w is a constant. So, in essence, guided local search tries to minimize a combination of the true cost and a penalty term. The weighting constant w is an important search parameter that determines how important the penalty term is with respect to the true cost.
Guided tabu search	Tabu search and guided local search can be mixed together. Arcs present in the solution are penalized just as with guided local search, while a tabu list is handled just as in tabu search. Therefore, the short-term memory feature of simple tabu search is enhanced with a long-term memory, acting as a diversifying scheme. This results in a very effective meta-heuristic, that can often produce good quality solutions in much fewer iterations than either simple Guided Local Search or Tabu Search.

In the prototype the savings heuristic and guided local search heuristic are used since they showed the best performance for the given task (these heuristics are italic in Table 20, Table 21).

5.7.2. Knowledge Sources for the Case Study

The experiments that require execution of the complete system scenario are based on the following initial data.

For the task of the routing plan creation the following KSs were considered:

- 9 available suppliers and constraints on suppliers' capabilities, capacities, locations are stored in an external database (3 types of suppliers, 3 alternative suppliers for each type); each supplier is characterized by its location, capabilities (which products it can produce), capacities (how many products it can produce by a certain moment).
- available providers of transportation services and constraints on available types, routes, and time of delivery are stored in an external database;
- factors influencing route availabilities as the geography and weather of the considered region and defining constraints on types, routes, and time of delivery, e.g. by air, by trucks, by off-road vehicles.

The latter are taken from two sources: from an external database and from a Web-site. For this purpose an emulated news Web site has been implemented (Figure 87). A specially designed wrapper reads news and finds which cities/areas are not currently available for transportation. Besides, it reads weather conditions and accordingly corrects transportation time and costs for appropriate routes.



Figure 87. Emulated news Web-sit

This list of sources is not fixed. The scalable architecture of the system KSNNet allows seamless attaching of new sources in order to get new features and to take into account more factors for tasks solved.

5.7.3. Implementation of Web-Service Interface

The example (Figure 89) refers to the mobile hospital configuration scenario described in the previous reports. It shows a `GetOperatingTablePrice` (SOAP/WSDL) request to an `OperatingTableQuote` service via the SOAP 1.1 HTTP binding. The request takes an application defined `TimePeriod` structure containing a start and end time for the operating tables production and returns an array of prices recorded by the service within that period of time, as well as the frequency at which they were recorded as the SOAP/WSDL response.

The user represented by a Web-service client enters a request into the system via the developed SOAP- based interface. It has been implemented using PHP (PHP, 2003) and NuSOAP Web Services Toolkit (NuSOAP, 2003). This combination enables rapid development of such applications as Web-services. Below, an example source code of the developed service is presented (Figure 88):

```
$s = new soap_server;
$s -> register('sendRequest');
$s -> register('getStatus');
$s -> register('getResult');
$s -> service($HTTP_RAW_POST_DATA);
```

Figure 88. Web service source code (example)

```

<?xml version="1.0"?>
<definitions name="StockQuote"
targetNamespace="http://example.com/stockquote.wsdl"
  xmlns:tns="http://example.org/operatingtable.wsdl"
  xmlns:xsd="http://www.w3.org/2000/10/XMLSchema"
  xmlns:xsd1="http://example.org/operatingtable/schema"
  xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
  xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
  xmlns="http://schemas.xmlsoap.org/wsdl/">

  <types>
    <schema targetNamespace="http://example.org/operatingtable/schema"
      xmlns="http://www.w3.org/2000/10/XMLSchema">
      <complexType name="TimePeriod">
        <all>
          <element name="startTime" type="xsd:timeInstant"/>
          <element name="endTime" type="xsd:timeInstant"/>
        </all>
      </complexType>
      <complexType name="ArrayOfFloat">
        <complexContent>
          <restriction base="soapenc:Array">
            <attribute ref="soapenc:arrayType" wsdl:arrayType="xsd:float[]" />
          </restriction>
        </complexContent>
      </complexType>
    </schema>
  </types>

  <message name="GetPricesInput">
    <part name="timePeriod" element="xsd1:TimePeriod"/>
  </message>

  <message name="GetPricesOutput">
    <part name="result" type="xsd1:ArrayOfFloat"/>
    <part name="frequency" type="xsd:float"/>
  </message>

  <portType name="OperatingTableQuotePortType">
    <operation name="GetLastPrice" parameterOrder="timePeriod frequency">
      <input message="tns:GetPricesInput"/>
      <output message="tns:GetPricesOutput"/>
    </operation>
  </portType>

  <binding name="OperatingTableQuoteSoapBinding" type="tns: OperatingTableQuotePortType">
    <soap:binding style="rpc" transport="http://schemas.xmlsoap.org/soap/http"/>
    <operation name="GetPrices">
      <soap:operation soapAction="http://example.org/GetPrices"/>
      <input>
        <soap:body use="encoded" namespace="http://example.org/operatingtable"
          encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" />
      </input>
      <output>
        <soap:body use="encoded" namespace="http://example.org/operatingtable"
          encodingStyle="http://schemas.xmlsoap.org/soap/encoding/" />
      </output>
    </operation>
  </binding>

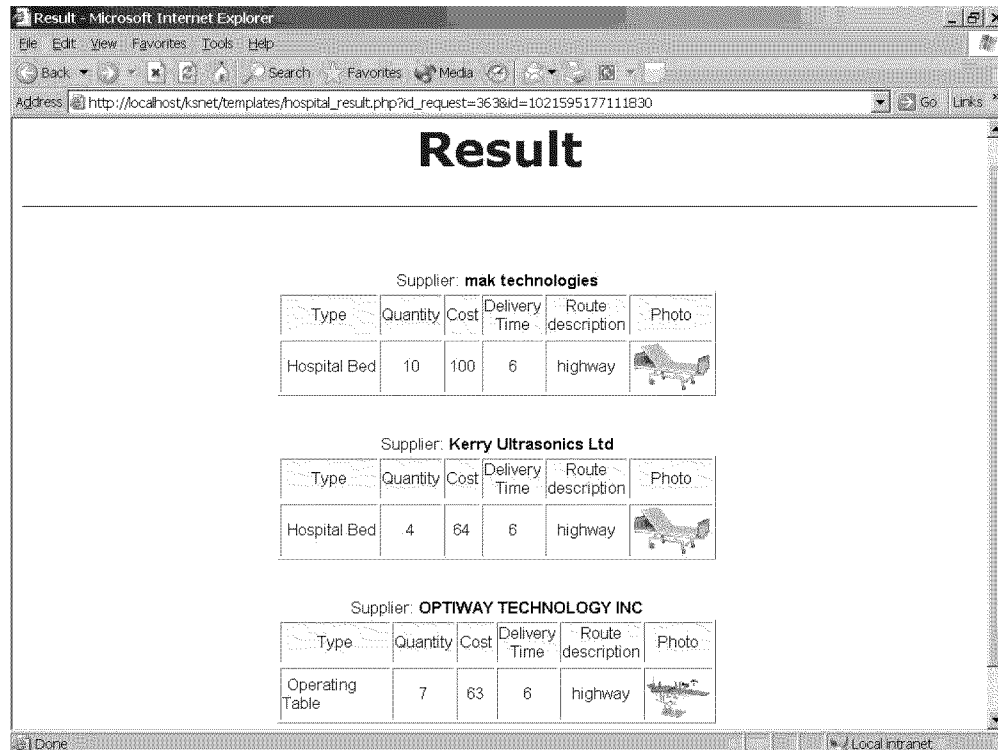
  <service name="OperatingTableQuoteService">
    <documentation>Operating Tables service</documentation>
    <port name="OperatingTableQuotePort" binding="tns: OperatingTableQuoteBinding">
      <soap:address location="http://example.org/operatingtable" />
    </port>
  </service>
</definitions><?xml version="1.0" encoding="utf-8" ?>

```

Figure 89. SOAP/WSDL document example


5.7.4. Experimentation with Different User Preferences

Presented example illustrates finding a routing plan for the same conditions but with different user preferences, namely: **minimize time**; **minimize time, then costs**; **minimize both time and costs**; **minimize costs, then time**; **minimize costs**. These preferences can be chosen on the request input screen of the system "KSNet". After the execution of the scenario the system shows the results (Figure 90). The activities of system's agents during the request processing are shown in (Figure 91).




Result

Supplier: **mak technologies**

Type	Quantity	Cost	Delivery Time	Route description	Photo
Hospital Bed	10	100	6	highway	

Supplier: **Kerry Ultrasonics Ltd**

Type	Quantity	Cost	Delivery Time	Route description	Photo
Hospital Bed	4	64	6	highway	

Supplier: **OPTIWAY TECHNOLOGY INC**


Type	Quantity	Cost	Delivery Time	Route description	Photo
Operating Table	7	63	6	highway	

Figure 90. Example result of supplier choice for user request

In (Figure 92 – Figure 94) results for different choices are presented and compared. For illustration of the results a map is generated that uses the following notations. Green dots are the cities of the region. The city with red edge (Aida) is the city where the hospital is to be located – the location of the customer. The cities with blue edges are the cities where suppliers are located (Libar, Higgville, Ugwulu, Langford, Nedalla, Laki, Dado). Transportations routes are shown as lines. The grey lines are routes that are not used for transportation in the solution, the blue lines routes used for transportation, and the red lines are routes unavailable due to weather or for some other reasons. E.g., the routes through the city of Zaribe are not available because of the flooding. The colored trucks denote the routes of particular vehicles/vehicle groups.

Figure 95 represents a comparison of the routing plans created for different criteria. As it can be seen while importance of one of the parameters increases (e.g., importance for costs increases from left to right) the value of the parameter decreases (the red line with diamonds for the costs) and vice versa (the green line with squares for the time).

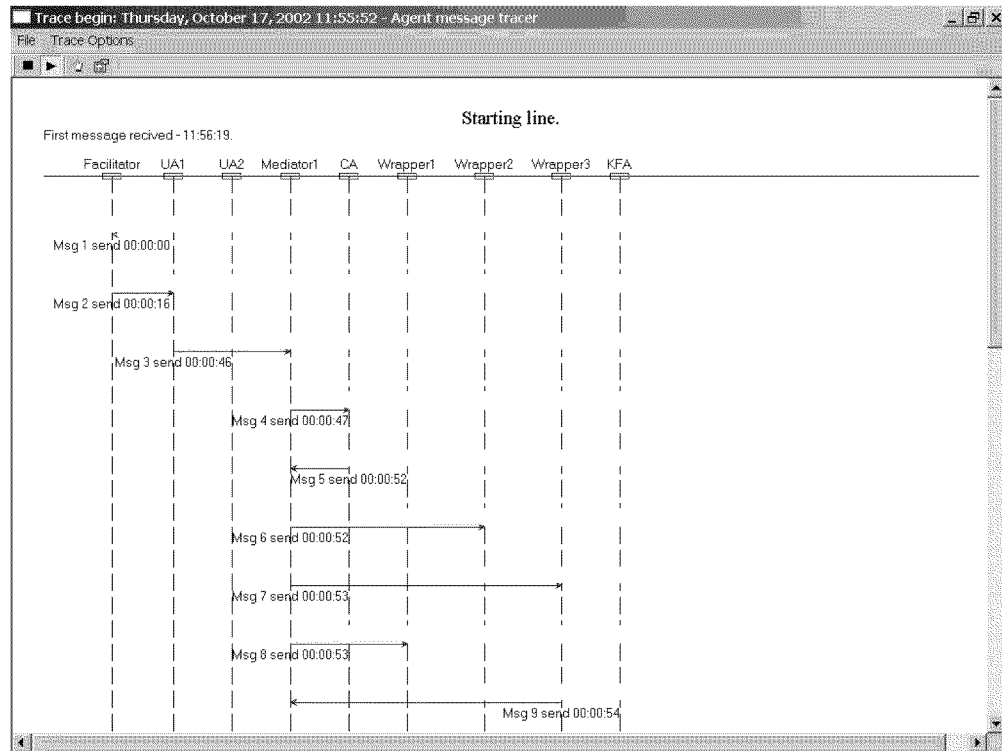


Figure 91. Example agent activity for user request

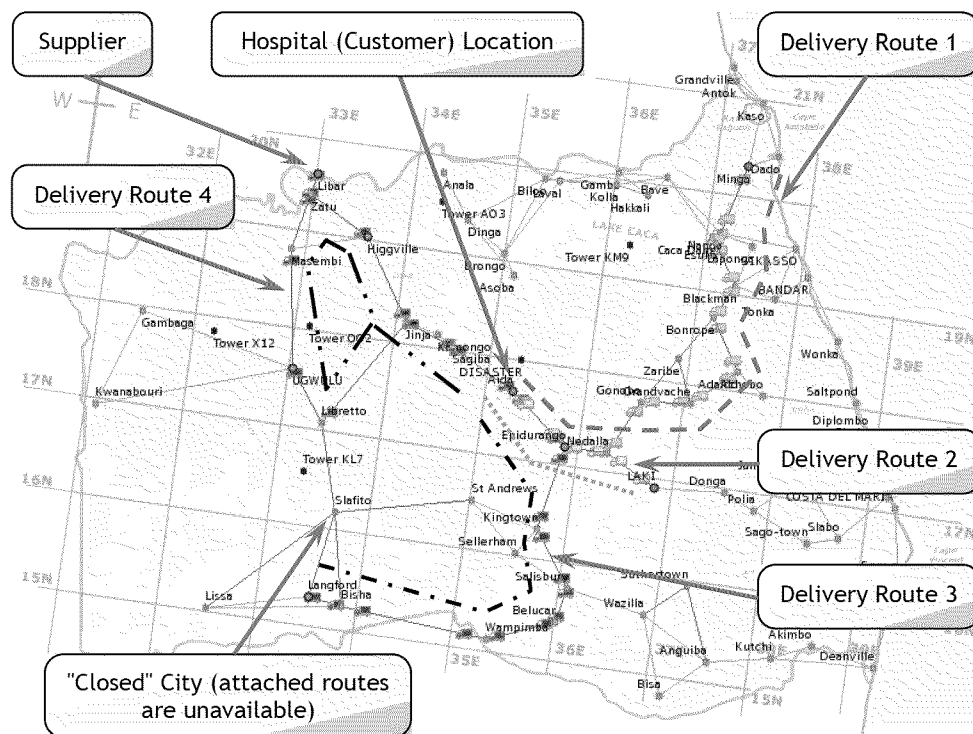


Figure 92. Routing plan for the minimize time and minimize time, then costs preferences (in this solution four vehicles/vehicle groups are used to provide maximum of concurrency)

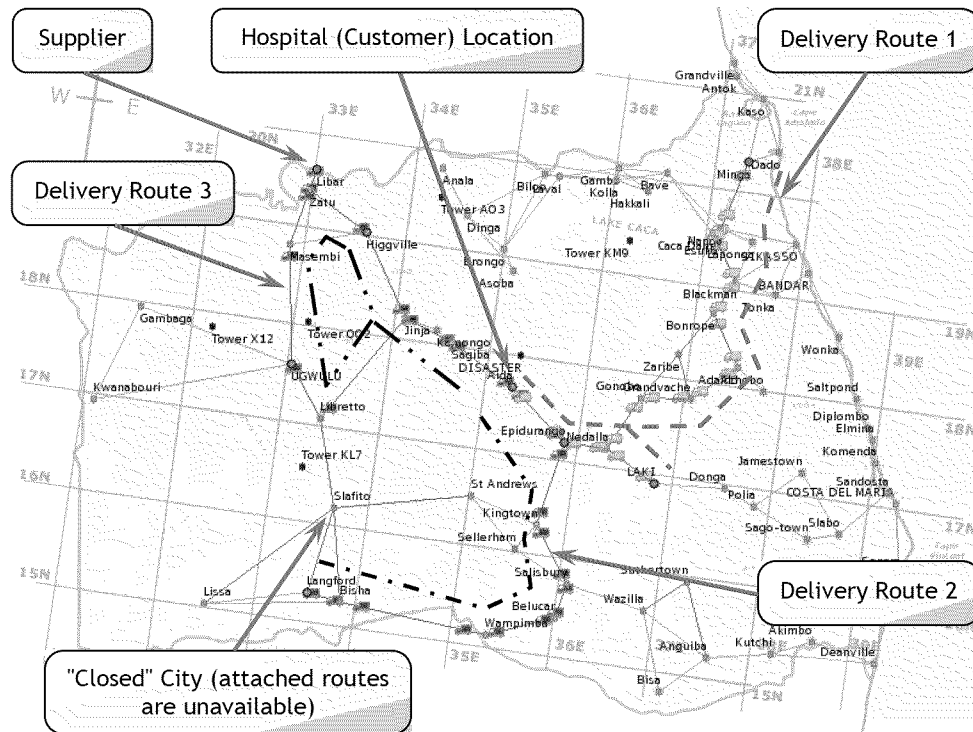


Figure 93. Routing plan for the minimize both time and costs and minimize costs, then time preferences (in this solution three vehicles/vehicle groups are used)

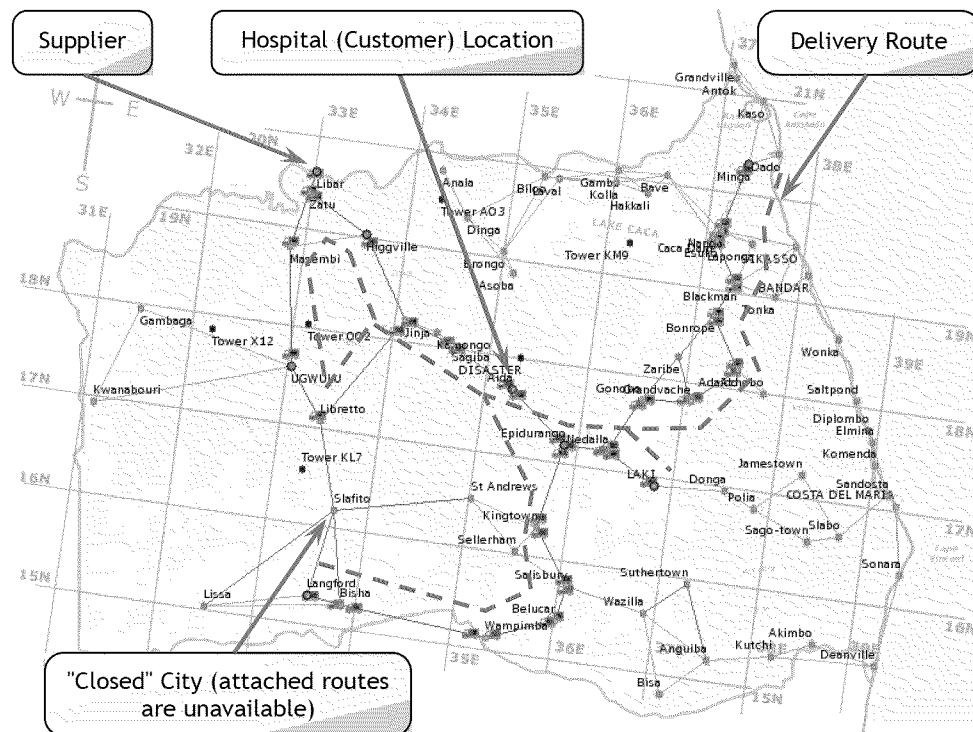


Figure 94. Routing plan for the minimize costs preference (in this solution one vehicle/vehicle group is used to provide minimum of costs)

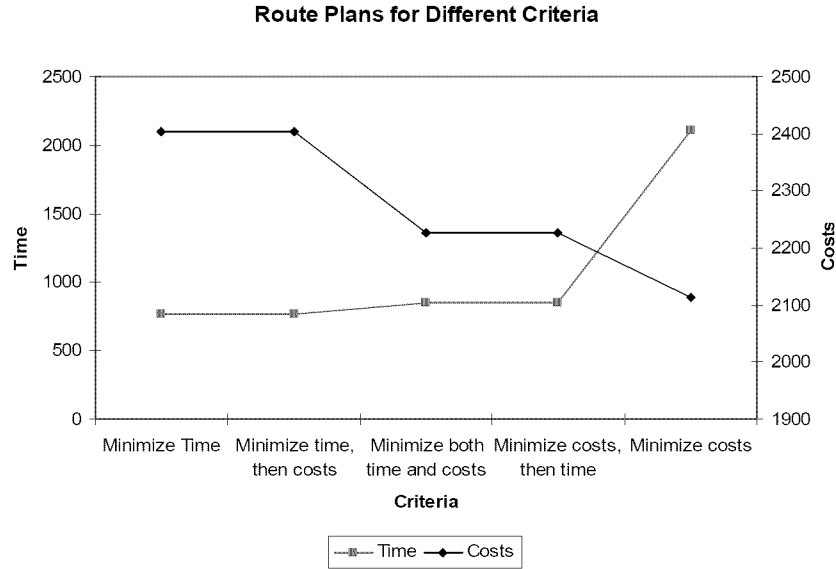


Figure 95. Routing plans for different criteria (time and costs minimization preferences)

5.7.5. Experimentation with Uncertainty Factors

5.7.5.1. Expert Estimation of a Constraint Importance

This example demonstrates the uncertainty caused by the assurance of experts in their knowledge. An expert group consisting of 7 experts estimated the importance of the constraint that defines if a certain city can be used for transportation or not (because of the weather or political situation).

In order to use this type of uncertainty in the user request processing the experts' opinions (0.6; 0.7; 0.5; 0.9; 0.6; 0.5; 0.8) have to be summarized. This is done via the following formulas:

$$\bar{\omega} = \sqrt[n]{\prod_{i=1}^n \omega_i}, \text{ where}$$

$\bar{\omega}$ – is the resulting value of the constraint importance,

ω_i – is the opinion of expert i ,

n – is the number of experts.

In the given example the resulting constraint importance is 0.64 (Figure 96).

5.7.5.2. Task Complexity Reduction via Omitting Constraint

This example is intended to demonstrate the influence of omitting a constraint on the task's solution. This is achieved by using uncertainties of the domain formalization. In the example the same case-study is considered but the constraint that defines if a certain city can be used for transportation or not (because of the weather or political situation) can be omitted. Its importance has been defined by the experts in the previous example is 0.64.

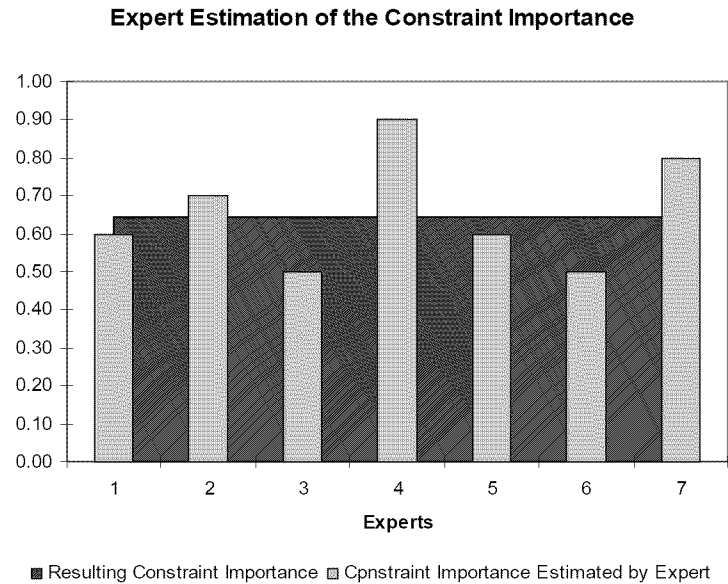


Figure 96. Expert estimation of the constraint importance

First the case with the omitted constraint is considered. In this case the solution generated assumes that all the cities are enabled for transportation (Figure 97). It has the following characteristics:

Costs:	2 207
Time:	682
Probability:	64%

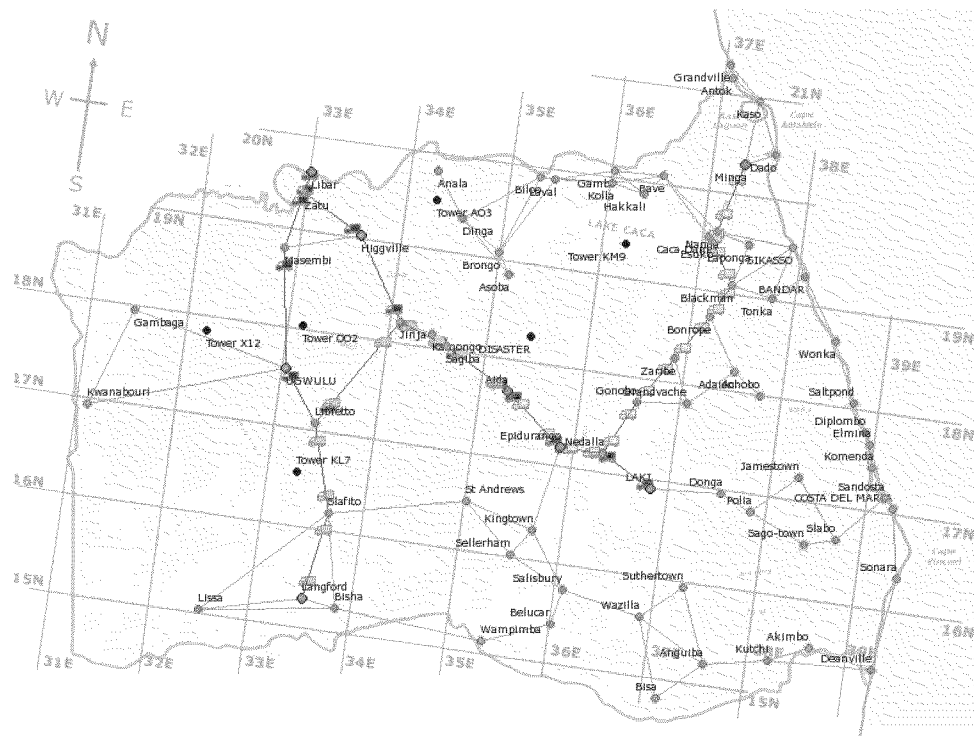


Figure 97. Routing plan when all cities are considered as available

The case where the constraint considered is not omitted results in the solution represented in (Figure 98). It has the following characteristics:

Costs: 5 845
 Time: 1626
 Probability: 100%

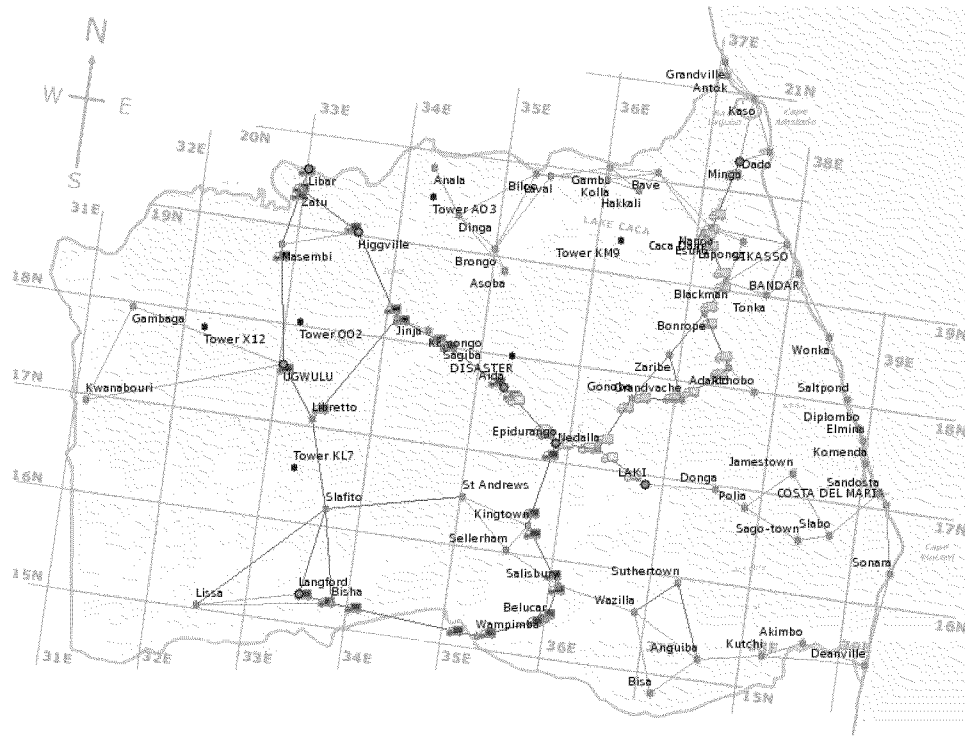


Figure 98. Routing plan when some of the cities are blocked

The experiments show that the solution obtained when the considered constraint is omitted is better in regard to its costs and time but has a low degree of assurance.

5.7.5.3. Uncertainty Presented in Knowledge Sources

Presented example illustrates treating uncertainties presented in KSs. The same case study as in the previous example is considered. However, in this example the degree of belief to the KS describing availability of the cities for transportation is limited. This KS is a simulated news Web-site (sec. 5.7.2) that publishes news about the Binni region. Particularly, it publishes warnings about some cities in case of negative weather or political conditions. The probability of a city to be blocked for transportation if it has negative conditions is considered as 30%.

The example results in two routing plans: one where all the cities are considered as available (Figure 97) and the other one when some of the cities are considered as blocked (Figure 98). The first result has the following characteristics:

Costs: 2 207
 Time: 682
 Probability: 24%

The second result has the following characteristics:

Costs: 5 845
 Time: 1626
 Probability: 75%

The experiments show that the solutions obtained when required information is available are optimistic with a very low degree of certainty of 24% and pessimistic with a degree of certainty of 75%.

5.7.5.4. Expert Estimation of Fuzzy Constraint

This example demonstrates a definition of a fuzzy constraint caused by the assurance of experts in their knowledge. The BOM constraint defining the number of operating tables per patient is considered. By default, it sets the amount of 5 operating tables per 50 patients (sec. 5.7.1.1) – the crisp constraint.

The experts suggested the values presented in (Table 22).

Table 22. Expert values of the number of operating tables per patient

Expert	Min	Max
1	2	8
2	4	6
3	1	7
4	3	6
5	4	8
6	2	7
7	3	7

The resulting fuzzy function is shown in (Figure 99) and can be formalized with the following trapezoidal function:

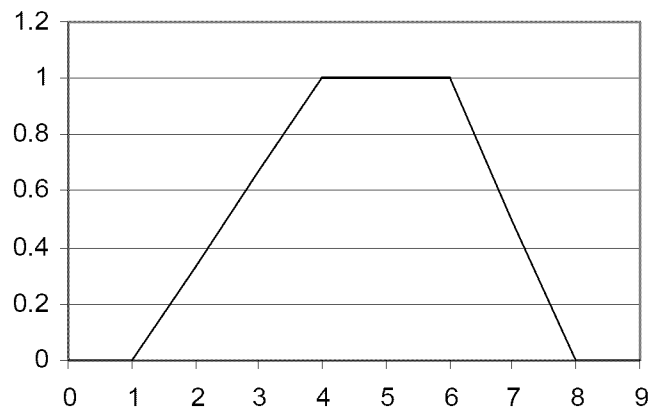


Figure 99. Expert estimation of the fuzzy constraint

$$\omega(c) = \begin{cases} 0, m \leq 1; m \geq 8 \\ \frac{m-1}{3}, 1 < m < 4 \\ 1, 4 \leq m \leq 8 \\ \frac{8-m}{2}, 6 < m < 8 \end{cases}, \text{ where}$$

$\omega(c)$ – fuzzy value of the constraint,
 m – number of operating tables per 50 patients.

5.7.5.5. Constraint Relaxation Using Fuzzy Constraints

This example is intended to demonstrate how a fuzzy constraint can be relaxed. This is achieved by using uncertainties of the domain formalization. In the example the constraint defining the number of operating tables per patient is considered.

In the example the capacities of suppliers were changed so that they could not provide the required number of operating tables for the hospital to be built. The hospital capacity is 500 patients and suppliers can produce only 30 operating tables.

In this case the constraint defining the number of operating tables per patient is relaxed so that instead of requirements of 5 operating tables per 50 patients only 3 are required. This leads to a decrease of the solution certainty level to 66.67%.

5.8. Agents' Load Analysis

For visualization of agents activities during different scenarios processing the Gantt chart diagrams were utilized. They allow presenting each agent's task as a bar on a time-based grid. Such diagrams help to define the agents most loaded on the various tasks.

If Figure 100 a Gantt chart diagram of agents' activities during template-based user request processing is presented. An analysis of a large amount of such diagrams for different scenarios has been performed. The results show that during user request processing the most of the time is spent by: (i) wrappers when they serve "slow" KSs, (ii) KF agent when it processes a large amount of constraints, (iii) translation agent when it recognizes long user requests and (iv) configuration agent when it finds a configuration of KSNet s consisting of a large number of KSs.

Combining Gantt chart diagrams with traditional pie and column diagrams (Figure 101) allows to see time distribution during different scenarios processing between members of the agents community. In the presented scenario of KSNet configuration the wrappers were responsible for the following sources: *wrapper 1*: geographic data acquisition; *wrapper 2*: weather and political conditions; *wrapper 3*: suppliers and related information.

As it can be concluded that most of the time is spent by wrappers for accessing KSs (since the Web-site is emulated the time of access to it is comparable with the time of accessing databases). The time spent by configuration agent for analysis of wrappers' proposals and making decision is small.

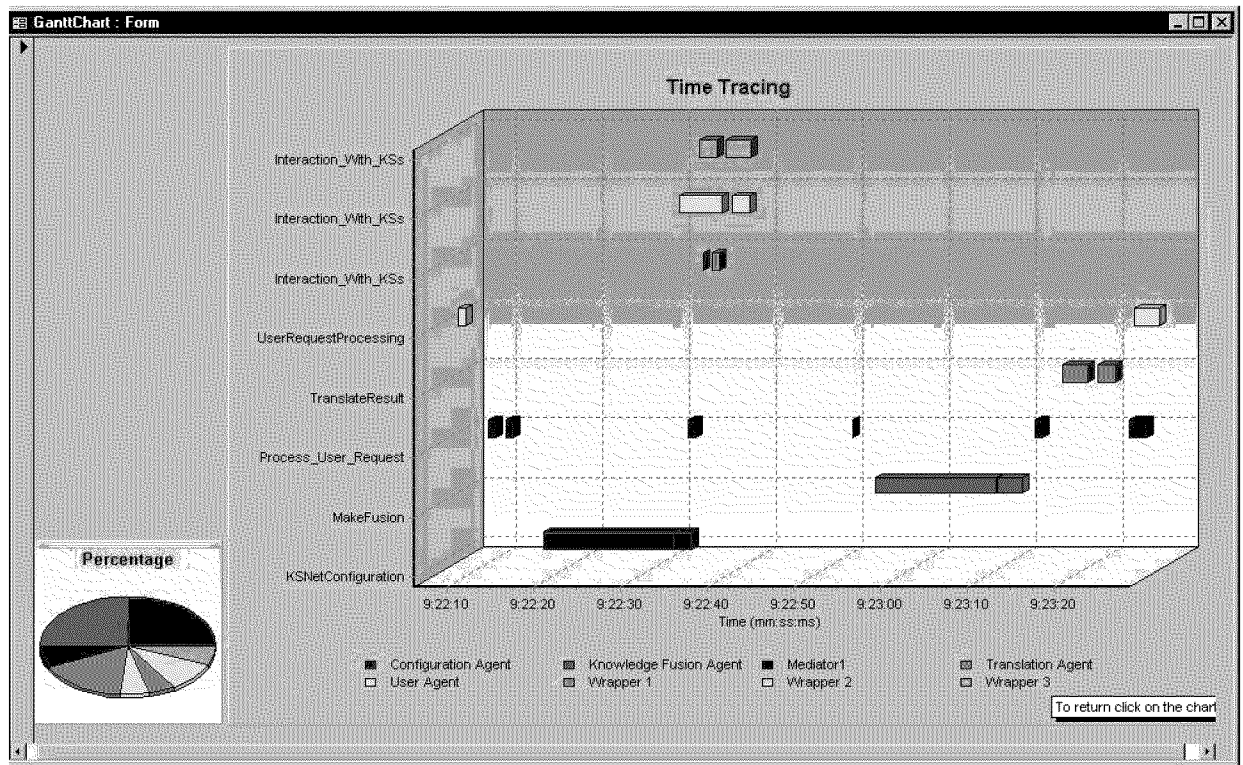


Figure 100. Visualization of agent activities during template-based user request processing

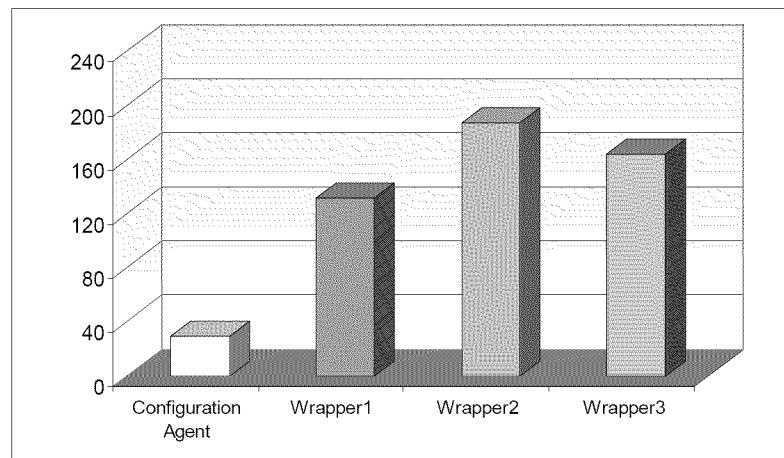


Figure 101. Example of column diagram for agents charge comparison

6. CONCLUSION AND FUTURE WORK

6.1. Summary of Results

The developed fusion-based knowledge logistics methodology can be a very powerful concept to enable collaboration between members of joint actions and operations. This concept can be applied to coalition-based operations in many industrial, healthcare, and other than war military applications featuring large-scale dynamic virtual organizations with distributed operations, logistics operations addressing end-to-end rapid supply, markets via partnerships, etc.

Below the main results of the project are presented.

Analysis of modern knowledge management system allowed to formulate advanced modern requirements to fusion-based knowledge management systems. Among them are: (i) flexibility, (ii) customizability, and (iii) open connectivity. There were revealed types of KSs located in the infosphere and selected common formats for knowledge processing: (i) passive sources (available external data- and knowledge- bases, structured documents, other sources with some developed mechanisms of interaction) providing knowledge by request; and (ii) active sources (experts, knowledge management tools) providing knowledge by request and pro-activity functions “Just-Before-Time”–support for request processing.

There were investigated operations of knowledge fusion, developed conceptual model of fusion-based knowledge logistics process. Consumer-focused ontology-driven methodology for knowledge logistics system implementation was developed: (i) conceptual model of fuzzy object-oriented constraint network was adapted for ontology representation formalism, and (ii) mathematical models of appropriate ontology operations were developed (Interim Report # 2, 2001). The ‘tacit-explicit model’ of knowledge sharing dealing with knowledge characteristics was adapted (Interim Report # 5, 2003). The Grid-based open service model was chosen as the system “KSNet” communication platform.

Integrated framework of fusion-base knowledge logistics into scalable infosphere was developed. It includes such components as a knowledge repository, a agents community and distributed expert groups. It uses such technologies as ontology management, intelligent agents, constraint satisfaction/propagation, soft computing, virtual reality and groupware.

A multiagent architecture for fusion-based knowledge logistics technology was developed. FIPA reference model was chosen as a kernel for agents community implementation. Technological and problem-oriented agents were selected, their functions and scenarios were presented using interaction diagram. Iterated Constraint-Based Contract Net Protocol was developed for agents’ negotiation.

A research software prototype that validates the major solutions concerning architecture, models and methods of above agent-based was implemented. UML-based conceptual projects describing the knowledge logistics system scenarios, agents’ behavior and major systems units were developed. A scenario of the system “KSNet” implementation as an open service was designed.

Case-based evaluation of the facilities of the developed integrated approach (methodology, integrated framework, and a set of models and methods) implemented within the prototype based on Binni scenario of coalition operations for health service logistics was done.

6.2. Conclusions Based on Research Prototype Evaluation

The major aims of developed research prototype were:

- to check advanced requirements, methodology and framework of rapid knowledge fusion in the scalable infosphere;
- to implement agent-based architecture for fusion-based knowledge logistics technology;
- to create interface forms and support procedures for direct knowledge entry by domain experts and for knowledge repository parallel development by distributed teams;
- to check developed models, agent architectures and prototypes for knowledge sharing by knowledge maps, and for distributed uncertain knowledge management.

All these tasks were implemented in the framework of the research prototype. The developed research prototype has demonstrated the following advantages of the proposed approach:

- Due to integration of constraint satisfaction technology with knowledge management approaches a new intelligent service for Knowledge Grid environment was implemented.
- The used in the system "KSNet" object-oriented constraint networks formalism is compatible with DAML+OIL knowledge representation format. This was checked by import of accessible in the Internet ontologies from DAML+OIL format.
- Integration of constraint satisfaction technology with selected knowledge representation allowed to implement a KF agent that generates a problem solving procedure for user request processing "on-the-fly". It enables reuse of developed module even if a user changes parametric constituent of his/her request or AO is modified. Distributed architecture of the KF agent allows to decompose complex tasks in subtasks and solve them concurrently.
- Usage of uncertainties in the KSNet-approach allows to find feasible solution where not all of the revealed constraints can be satisfied.
- Integration of linguistic and numeric information facilitates insertion of expert knowledge in the system "KSNet" knowledge repository.
- Usage of a free text recognition module increases rapidity of the user request processing due to the following:
 - Misspelled words can be found at the beginning of request processing;
 - Correspondence between words from user request and ontology elements can be determine;
 - Due to usage of regular expression a parametric constituent from a user request can be found.
- Usage of task and method ontologies allowed to define value of different application ontology attributes in accordance with user defined constraints found in a user request.

The developed prototype can be demonstrated by request.

6.3. Possible Directions of the Project Evolution

Grid is a modern technology for parallel distributed system development that enables sharing, selection, and aggregation of resources distributed across "multiple" administrative domains based on their availability, capability, performance, cost, and users' quality-of-service requirements (Grid definition, 2003). **Knowledge Logistics** tightly correlates with the ideas of Semantic Grid and can be considered **as an Intelligent Service for Knowledge Grid** environment (Figure 102). Most of the current research efforts are devoted to development of Information and Data Computation services in the Information Grid environment. In the nearest future a shift to the global e-Application environment is expected. The developed technology of knowledge logistics advances into this new level of information technology.

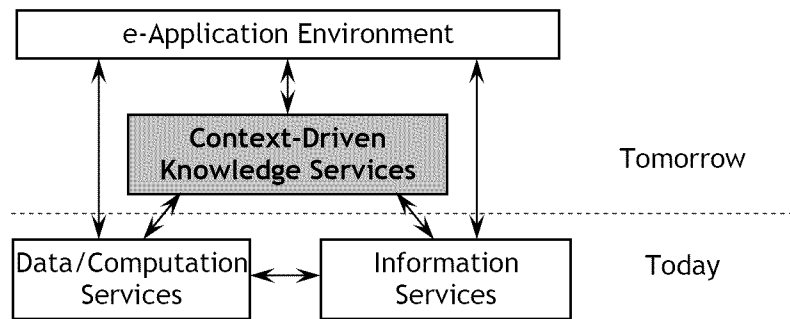


Figure 102. Future Work: Knowledge Service in Grid Environment

Today an efficient management of the information between multiple participating parties is needed to plan and predict results of any coalition operation. This information must be pertinent, clear, and correct, and it must be timely processed and delivered to appropriate locations, so that it could provide situation awareness and prediction. Without this level of efficiency it is hardly possible to predict developments or foresee problems in the contemporary everchanging and increasingly complex world. This would result in, for example, possible evolution of the current project as **agent-based information fusion support for on-the-fly coalition operation impact assessment in network-centric environment**. Here proposed project is intended to use major results of the KSNet-approach and to extend the KSNet-approach as shown in Figure 103 with orientation to on-the-fly impact assessment of coalition operations. The project could facilitate the process of decision making for **rapid response coalition operation impact assessment** through information fusion at levels 2+ and 3 of JDL (“Joint Director of Laboratories”) fusion model.

Examples of rapid response coalition operations include:

Operations Other than War (OOTW) such as peace-keeping, peace-enforcing, non-combatant evacuation or disaster relief operations, are very likely based on the cooperation of a number of different coalitions such as quasi-volunteer, vaguely organized groups of people, non-government organizations, institutions providing humanitarian aid, and also military personnel, and official governmental initiatives.

Operations supporting anti-terrorism as well as *homeland security and defense systems*, consisting of hierarchically organized military and official governmental personnel and also involving non-military organizations.

Protection of public service systems such as power grids, water-supply, 911 system, etc.

Logistics operations management for disaster relief addressing sustainment, transportation and end-to-end rapid supply to the final destination.

New project assumes change and extension of the existing multiagent architecture of the KSNet-approach and development of new advanced machine learning models that will allow modeling and analysis of complex "what-if" scenarios of coalition operations in a network-centric environment. It is also planned to use such technologies as computer-aided reasoning and soft computing.

On-the-fly impact assessment is based on the analysis of context retrieved from different units of infosphere (users and knowledge sources). Context is any information that can be used to characterize the situation of an entity where an entity is a person, place, or object that is considered relevant to the interaction between a user and an application, including the user and applications themselves (Dey et al., 2001). The efforts of combining accumulated knowledge to reuse it (including efforts of knowledge fusion) have led to a necessity to develop methods for knowledge content retrieval and analysis. Representation of user preferences and knowledge source features by the used ontology formalism gives a possibility to obtain context-dependent application ontologies. Versioning these ontologies and support of reasoning about them enable context retrieval. Context analysis is based on machine learning technique. All this tasks lead to usage of ontology learning - a modern scientific

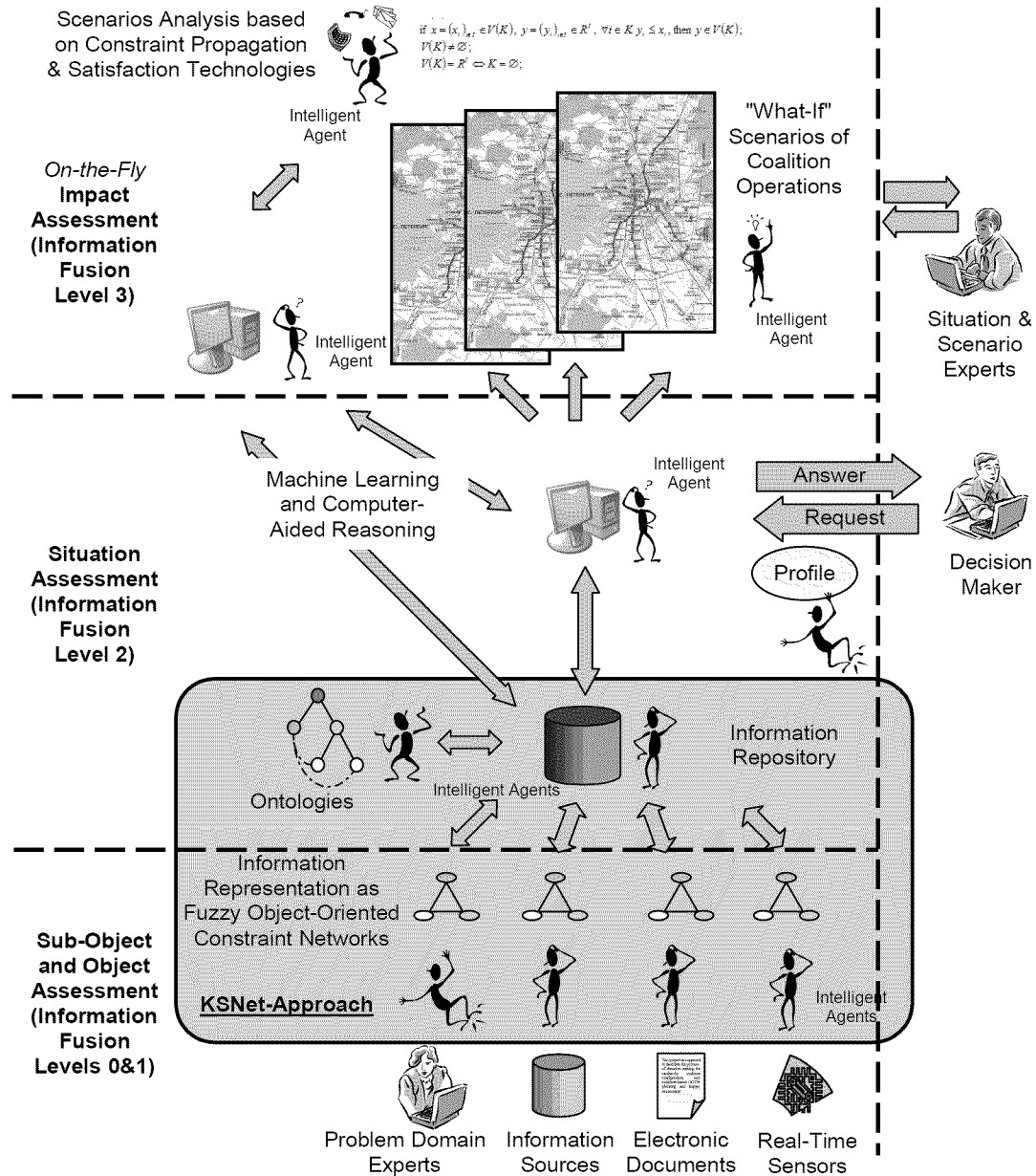


Figure 103. Technologies to be used in the proposed approach to information fusion

direction in the ontology management. It can be defined as ontology engineering using machine learning methods. Research works published in Semantic Web community (Kaikova et al., 2000; Kavalec and Svatek, 2002) define the following 5 stages of ontology learning: (i) import and reuse, (ii) extraction, (iii) pruning, (iv) refinement, and (v) validation (the prime target application serves as a measure for validating the resulting ontology). Usage of the developed ontology-driven methodology in this area could make valuable contributions in this area since all these operations were described and implemented in the project.

6.4. Publications

Total number of publications with the main project results is 34. Below is presented a list of selected publications.

International Journals

- Smirnov A., Sheremetov L., Chilov N., Cortes J.R. Soft-computing technologies for configuration of cooperative supply chain. *International Journal on Applied Soft Computing*. (Accepted for publication in November, 2003).
- Smirnov A., Pashkin M., Chilov N., Levashova T. Knowledge Logistics in Information Grid Environment. *The special issue "Semantic Grid and Knowledge Grid: The Next-Generation Web" of International Journal on Future Generation Computer Systems*. (Accepted for publication in April, 2003).
- Smirnov A., Pashkin M., Chilov N., Levashova T. Haritatos F. Knowledge Source Network Configuration Approach to Knowledge Logistics. *International Journal of General Systems*. Taylor & Francis Group, 2003, 32 (3), pp. 251—269.
- Smirnov A., Pashkin M., Chilov N., Levashova T. Agent-Based Support of Mass Customization for Corporate Knowledge Management, *Engineering Applications of Artificial Intelligence*. Volume 16, Issue 4, June 2003, pp. 349—364.
- Smirnov A., Pashkin M., Chilov N., Levashova T. KSNet-Approach to Knowledge Fusion from Distributed Sources. *Computing and Informatics*. V. 22, 2003, pp. 105—142.

Book chapters

- Smirnov A., Pashkin M., Chilov N., Levashova T. Krizhanovsky A. In: R. Meersman, Z. Tari, D.C. Schmidt et al. (Eds.): *Ontology-Driven Knowledge Logistics Approach as Constraint Satisfaction Problem. On the Move to Meaningful Internet Systems 2003. Lecture Notes in Computer Science 2888*. Springer, 2003. pp. 535—652.
- Smirnov A., Pashkin M., Chilov N., Levashova T. Multi-Agent Knowledge Logistics System “KSNet”: Implementation and Case Study for Coalition Operations. In: V. Mařík, J. Müller, M. Pechouček (Eds.): *Multi-Agent Systems and Applications. Lecture Notes in Artificial Intelligence 2691*, Springer-Verlag, Berlin, Heidelberg, 2003. pp. 292—303.
- Smirnov A., Pashkin M., Chilov N., Levashova T. Knowledge Fusion in the Business Information Environment for e-Manufacturing Pursuing Mass Customisation. *Moving into Mass Customization. Information Systems and Management Principles*. (eds. by C. Rautenstrauch, R. Seelmann-Eggebert, K. Turowski), Berlin: Springer, 2002. pp. 153—175.
- Smirnov A., Pashkin M., Chilov N., Levashova T. Multi-agent Architecture for Knowledge Fusion from Distributed Sources. *Lecture Notes in Artificial Intelligence 2296*. Springer, 2002. pp. 293—302.

Proceedings of International Conferences

- Smirnov A., Pashkin M., Chilov N., Levashova T. Agent-Based Knowledge Logistics for Coalition Operations: Main Technologies and a Case Study. *Proceedings of the 2003 IEEE International Conference on Information Reuse and Integration*. Las Vegas, USA, October, 27—29, 2003. IEEE Systems, Man, and Cybernetics Society, 2003. 609—616.
- Smirnov A., Pashkin M., Chilov N., Levashova T. ‘Knowledge Source Network Configuration in e-Business Environment’. *Proceedings of the 15th IFAC World Congress (IFAC’2002)*. Barcelona, Spain, July, 21-26, 2002. (Electronic Proceedings).
- Smirnov A., Pashkin M., Chilov N., Levashova T., Haritatos F. A KSNet-Approach to Knowledge Logistics in Distributed Environment. *Proceedings of the International Conference on Human-Computer Interaction in Aeronautics (HCI-Aero 2002)*. USA, Menlo Park: AAAI Press, October, 23—25, 2002. P. 88—93.
- Smirnov A., Pashkin M., Chilov N., Levashova T. ‘Knowledge Source Network Configuration Approach to Knowledge Logistics’. *Proceedings of the IEEE International Conference on Artificial Intelligence Systems (ICAIS’02)*. Divnomorskoe, Russia, September, 5-10, 2002, pp. 95—100.

- Smirnov A., 2001. 'Profile-based configuring of knowledge supply networks in the global business information environment'. *Proceedings of the 2001 IEEE International Conference on Systems, Man and Cybernetics "e-Systems and e-Man for Cybernetics in Cyberspace"*. Tucson, Arizona, October, 7-10, 2001. pp. 977—982.

Among the conferences major results of the project were presented the following can be outlined:

- The International Conference on Ontologies, Databases and Applications of Semantics Information Reuse and Integration (ODBASE'2003). Catania, Sicily, Italy, November, 3—7, 2003.
- The 2003 IEEE International Conference on Information Reuse and Integration. Las Vegas, USA, October, 27—29 2003.
- The Third e-Business Research Forum (eBRC'2003), Tampere, Finland, September, 23—25, 2003.
- The International Workshop on Information Fusion and Geographic Information Systems, St.Petersburg, Russia, September, 17—20 2003.
- International ISTC Seminar on Science and Computing. Moscow, Russia, September, 15—17, 2003.
- Data Fusion for Situation Monitoring, Incident Detection, Alert and Response Management, NAREK center of Yerevan University, Tsakhkadzor, Armenia, August, 18—29, 2003.
- The 10th International Conference on Concurrent Engineering – The Vision for the Future Generation in Research and Applications (ISPE'2003). Madeira, Portugal, July, 26—30, 2003.
- The 3rd International Central and Eastern European Conference on Multi-Agent Systems (CEEMAS 2003). Prague, Czech Republic, June, 16—18, 2003.
- The 9th International Conference on Concurrent Enterprising (ICE 2003). Espoo-Helsinki, Finland, June, 16—18, 2003.
- The 5th International Conference On Enterprise Information Systems. Angers, France, April, 23—26 2003.
- International Conference on Intelligent Manufacturing Systems, Budapest, Hungary, April, 6—8, 2003.
- The 9th International Conference on Neural Information Processing; 4th Asia-Pacific Conference on Simulated Evolution and Learning; 2002 International Conference on Fuzzy Systems and Knowledge Discovery (ICONIP'02-SEAL'02-FSKD'02). Singapore, November, 18—22, 2002.
- E-Business Research Forum (eBRF'2002). Tampere, Finland, November 14—16, 2002.
- International Conference on Human-Computer Interaction in Aeronautics (HCI-Aero 2002). Massachusetts, MA, USA, October, 23—25, 2002.
- The 9th International Multi-Conference on Advanced Computer Systems (ACS'2002). Szczecin, Poland, October, 23—25, 2002.
- The IEEE International Conference on Artificial Intelligence Systems (ICAIS'02). Divnomorskoe, Russia, September, 5—10, 2002.
- The 15th IFAC World Congress (IFAC'2002). Barcelona, Spain, July, 21—26, 2002.
- The 5th International Conference on Information Fusion. Annapolis, MD, USA. July, 6 – 12, 2002.
- The Eighth International Conference on Concurrent Enterprising (ICE 2002). Rome, Italy; June, 17—19, 2002.
- The Sixth International Research Conference on Quality, Innovation & Knowledge Management (QIK'2002). Kuala Lumpur, Malaysia. February, 17—20, 2002.
- The First International ICSC Congress on Autonomous Intelligent Systems (ICAIS'2002). Deakin University, Geelong, Australia; February, 12-15, 2002.
- IEEE International Conference on Systems, Man and Cybernetics "e-Systems and e-Man for Cybernetics in Cyberspace". Tucson, Arizona, USA, October, 7-10, 2001.
- The Second International Workshop of Central and Eastern Europe on Multi-Agent Systems (CEEMAS'2001). Krakow, Poland, September, 26-29, 2001.

REFERENCES

- (ActivePerl, 2002) ActivePerl. The industry-standard Perl distribution for Linux, Solaris, and Windows, ActiveState, 2002. <http://www.activestate.com/Products/ActivePerl/>.
- (ActiveState, 2002) ActiveState Homepage, ActiveState, 2002. <http://www.activestate.com/>.
- (Agent projects, 2002) Agent projects. Europe's Network of Excellence for Agent-based Computing. <http://www.agentlink.org/resources/agentprojects-db.html>.
- (Agha, 1986) Agha G. Actors: A Model of Concurrent Computation in Distributed Systems. Cambridge, MA: MIT Press, 1986.
- (Andreoli et al., 1994) Andreoli J.-M., Borghoff U.M., and Pareschi R. Constraint-based knowledge brokers. *Proceedings of the 1st International Symposium on Parallel Symbolic Computation (PASCO'94)* (H.Hong, ed.). Hagenberg/Linz, Austria, September 1994. Lecture Notes Series in Computing. Singapore, New Jersey, London, Hong Kong: World Scientific. **5**, 1—11.
- (Andreoli et al., 1995) Andreoli J., Borghoff U., and Pareschi R. Constraint Agents for the Information Age. *Journal of Universal Computer Science*, **1**, 1995. 762—789.
- (Andreoli et al., 1996) Andreoli J., Borghoff U., and Pareschi R. The Constraint-Based Knowledge Broker Model: Semantics, Implementation and Analysis. *Journal of Symbolic Computation*, **21**, 4, 1996. 635—667.
- (Anken, 2002) Anken C.S. Information Understanding. Introduction for 5th Anniversary Information Workshop. Rome, NY. The Information Institute. Air Force Research Laboratory (AFRL), Information Directorate. 2002. URL: http://www.rl.af.mil/rrs/Info_Inst/docs/briefings_2002/IU_II_workshop.ppt.
- (Blaxxun, 2002) Blaxxun Interactive 2002. <http://www.blaxxun.com>.
- (Bratman et al., 1988) Bratman M., Israel D., and PoUack M. Plans and resource-bounded practical reasoning. *Computational Intelligence*, **4**, 1988. 349—355.
- (Broll, 1997) Broll, W. 1997. Distributed Virtual Reality for Everyone – a Framework for Networked VR on the Internet. IEEE Virtual Reality Annual International Symposium (VRAIS'97), IEEE Computer Society Press, 121-128.
- (Brooks, 1991) Brooks R. Intelligence without representation. *Artificial Intelligence*, **47**, 1991. 139—159.
- (Brustoloni, 1991) Brustoloni J.C. Autonomous Agents: Characterization and Requirements. Technical Report CMU-CS-91-204, School of Computer Science, Carnegie Mellon University, November 1991.
- (Chaudhri et al., 1998) Chaudhri V.K., Farquhar A., Fikes R., Karp P.D., Rice J.P. Open Knowledge Base Connectivity Specification. Version 2.0.3. Stanford University, USA, 1998. <http://www-ksl-svc.stanford.edu:5915/doc/release/okbc/okbc-spec/index.html>.
- (Chaudhri et al., 2000) V.K. Chaudhri, J.D. Lowrance, M.E. Stickel, J.F. Thomere, and R.J. Wadlinger. Ontology Construction Toolkit (Technical Note Ontology, AI Center, Report, January 2000) SRI Project No. 1633. 85.
- (Clin-Act, 2000) Clin-Act (Clinical Activity), The ON9.3 Library of Ontologies: Ontology Group of IP-CNR (a part of the Institute of Psychology of the Italian National Research Council (CNR)), December, 2000. <http://saussure.irmkant.rm.cnr.it/onto/>.
- (Cyc, 1998) Hpkb-Upper-Level-Kernel-Latest: Upper Cyc/HPKB IKB Ontology with links to SENSUS, Version 1.4, February 1998. Ontolingua Ontology Server. <http://www-ksl-svc.stanford.edu:5915>.
- (DAML+OIL, 2001) Reference description of the ontology markup language, eds. by van Harmelen F., Horrocks I., 2001. URL: <http://www.daml.org/2001/03/reference>
- (DARPA, 2001) DARPA Advanced Logistics Project, 2001. www.darpa.mil/iso/alp.
- (Davis and Smith, 1983) Davis R. and Smith R.G. Negotiation as a metaphor for distributed problem solving. *Artificial Intelligence*, 1983, Vol. 20, No. 1. 63—109.
- (DeepMatrix, 2002) DeepMatrix, Geometrek 2002. URL: <http://www.geometrek.com/products/deepmatrix.html>
- (Desachy, Roux, and Zahzah, 1996) Desachy J., Roux L., Zahzah E.-H. Numeric and Symbolic Data Fusion: A Soft Computing Approach to Remote Sensing Images Analysis. *Pattern Recognition Letters*, 1996, 17. 1361—1378.
- (Dey et al., 2001) Dey A.K., Salber D., Abowd G.D. A Conceptual Framework and a Toolkit for Supporting the Rapid Prototyping of Context-Aware Applications. Context-Aware Computing. - A Special Triple Issue of Human-Computer Interaction / Eds. by T.P. Moran, P. Dourish. - Lawrence-Erlbaum, 2001. - Vol. 16 - <http://www.cc.gatech.edu/fce/ctk/pubs/HCIJ16.pdf>.
- (Durfee, 2001) Edmund H. Durfee: Distributed Problem Solving and Planning. EASSS 2001: 118-149
- (Filos and Ouzounis, 2001) Filos E., Ouzounis V. Virtual Organisations: Technologies, Trends, Standards and the Contribution of the European RTD Programmes. *International Journal of Computer Applications in Technology*, to appear in 2001. Special Issue: Applications in Industry of Product and Process Modeling Using Standards.
- (FIPA, 2002) FIPA – Foundation for Intelligent Physical Agents, 2002. <http://www.fipa.org>.
- (FIPA-OS, 2002) FIPA-OS toolkit. <http://fipa-os.sourceforge.net/>

- (FOLDOC, 2003) Free On-line Dictionary of Computing. <http://foldoc.doc.ic.ac.uk/foldoc/index.html>.
- (Gasser, 2001) Gasser, Les, Perspectives on Organizations in Multi-Agent Systems, Multi-Agent Systems and Applications, eds. Michael Luck et al., Springer-Verlag, Berlin, 2001, pp. 1-16.
- (Giachetti, et. al., 1997) Giachetti R., Young R., Roggatz A, Eversheim W., and Perrone G. A methodology for the reduction of imprecision in the engineering process. *European Journal of Operational Research*, 100, 1997. 277—292.
- (GNU, 2003) GNU Project web-site, 2003. URL: <http://www.gnu.org>.
- (Gorodetski, et. al., 2001) Gorodetski V., Karsayev O., Kotenko I., Khabalov A. Software Development Kit for Multi-agent Systems Design and Implementation. *Proceedings of the Second International Workshop of Central and Eastern Europe on Multi-agent Systems (CEEMAS'01)*. (B.Dunin-Keplicz, E.Nawarecki (eds.)). Krakow, Poland, 26-29 September, 2001. 99—108.
- (Grid definition, 2003) The Future of Distributed Middleware: Grid Computing and Web Services, SIMC's General Meeting (November, 2002) URL: <http://www.simc-inc.org/archive0203/Grid/>.
- (Guarino, 1998) Guarino N. Formal Ontology and Information Systems. Proceedings of FOIS'98. Trento, Italy. Amsterdam: IOS Press, 1998. 3—15.
- (Holsapple and Singh, 2001) Holsapple C. V., Singh M. The Knowledge Chain Model: Activities for Competitiveness. *Expert System with Applications*, 2001. 20. 77—98.
- (Hunter, 2000) Hunter A. Merging Potentially Inconsistent Items of Structured Text. *Data & Knowledge Engineering*, 2000. 305—332.
- (IBM, 2003) IBM corporate web-site, 2003. URL: <http://www.ibm.com>.
- (ILOG, 2003) ILOG Corporate Web-site, 2003. URL: <http://www.ilog.com>.
- (ILOG, 2003) ILOG Corporate Web-site, 2003. URL: <http://www.ilog.com>.
- (Intel Corp., 1998) Intel Corporate Press Releases, 1998. March, 4. <http://www.itanium.ru/pressroom/archive/releases/-CN030498.HTM>.
- (Intelligent Agents, 2002) Intelligent Agents. ISR Agent Research. <http://193.113.209.147/projects/agents.htm>
- (Interim Report # 1, 2001) Project # 1993P. Mathematical Basis of Knowledge Discovery and Autonomous Intelligent Architectures. Task # 2: Rapid Knowledge Fusion in the Scalable Infosphere. Interim Report # 2, St.Petersburg, 2001.
- (Interim Report # 2, 2001) Project # 1993P. Mathematical Basis of Knowledge Discovery and Autonomous Intelligent Architectures. Task # 2: Rapid Knowledge Fusion in the Scalable Infosphere. Interim Report # 2, St.Petersburg, 2001.
- (Interim Report # 3, 2002) Project # 1993P. Mathematical Basis of Knowledge Discovery and Autonomous Intelligent Architectures. Task # 2: Rapid Knowledge Fusion in the Scalable Infosphere. Interim Report # 2, St.Petersburg, 2002.
- (Interim Report # 4, 2002) Project # 1993P. Mathematical Basis of Knowledge Discovery and Autonomous Intelligent Architectures. Task # 2: Rapid Knowledge Fusion in the Scalable Infosphere. Interim Report # 4, St.Petersburg, 2002.
- (Interim Report # 5, 2003) Project # 1993P. Mathematical Basis of Knowledge Discovery and Autonomous Intelligent Architectures. Task # 2: Rapid Knowledge Fusion in the Scalable Infosphere. Interim Report # 5, St.Petersburg, 2003.
- (JADE, 2002) Java Agent DEvelopment Framework (JADE). <http://sharon.cselt.it/projects/jade/>
- (JP 4-02.1, 1997) JTTP for Health Service Logistic Support in Joint Operations, 1997. http://www.dtic.mil/doctrine/jel/-new_pubs/4_02_1.pdf.
- (Kaelbling and Rosenschein, 1990) Kaelbling L. and Rosenschein S. Action and planning in embedded agents. *Designing Autonomous Agents* (P.Maes, ed.), Cambridge, MA: MIT Press, 1990. 35—48.
- (Kaikova et al., 2000) Kaikova H., Terziyan V., Omelayenko B. Recognizing Bounds of Context Change in On-Line Learning. In: Proceedings of the Information Resources Management Association International Conference IRMA-2000, Anchorage, Alaska, USA, May 21-24 2000, IDEA Group Publishing (IGP), pp. 236-239.
- (Kavalec and Svatek, 2002) Kavalec M., Svatek V. Information Extraction and Ontology Learning Guided by Web Directory. In: ECAI Workshop on NLP and ML for Ontology engineering (OLT-02). Lyon, 2002.
- (KIF, 1992) Knowledge Interchange Format (ed. by M.R. Genesereth and R.E. Fikes). Version 3.0. Reference Manual, 1992. URL: <http://logic.stanford.edu/kif/Hypertext/kif-manual.html>.
- (Loom, 1997) Loom ontology browser, Information sciences Institute, The University of Southern California, 1997. <http://sevak.isi.edu:4676/loom/shuttle.html>.
- (Lytinen et al., 2000) Lytinen S., Tomuro N., and Repede T. The Use of WordNet Sense Tagging in FAQFinder. Proceedings of the Workshop on Artificial Intelligence for Web Search at the 17th National Conference on Artificial

- Intelligence (AAAI-2000), Austin, 2000.
- (Massive Multiagent Kit , 2002) Massive Multiagent Kit. Version 2.1. <http://www.gsigma-grucon.ufsc.br/massyve/mkit.htm>
- (Microsoft, 2003) Microsoft corporate web-site, 2003. URL: <http://www.microsoft.com>.
- (Miller et al., 1990) Miller G.A., Beckwith R., Fellbaum C., Gross D., and Miller K.J. Introduction to WordNet: an on-line lexical database. *International Journal of Lexicography*, 1990, 3(4), 235—244. <ftp://ftp.cogsci.princeton.edu/pub/wordnet/5papers.ps>
- (Miller, 1995) Miller A.G. Wordnet: A lexical database for English. *Communications of the ACM*, 38 (11), November 1995, 39—41. 1995. <http://www.cogsci.princeton.edu/~wn/>.
- (MTS, 1997) Glossary of Microsoft Transaction Server (MTS). Microsoft Corp., 1997. <http://www.fintech.ru/iishelp/mts/html/vipdef01.htm#asdefactivex>
- (NAICS, 2001) North American Industry Classification System code, DAML Ontology Library, Stanford University, July 2001. <http://opencyc.sourceforge.net/daml/naics.daml>
- (NuSOAP, 2003) NuSOAP Web Services Toolkit (2003). URL: <http://dietrich.ganx4.com/nusoap>.
- (Oltramari et al., 2002) Oltramari A., Gangemi A., Guarino N., Masolo C. Restructuring WordNet's Top-Level: The *OntoClean* approach. *Proceedings of LREC2002 (OntoLex workshop)*. Las Palmas, Spain, 2002.
- (Oussalah, 2000) Oussalah M. Study of Some Algebraical Properties of Adaptive Combination Rules. *Fuzzy Sets and Systems*, 2000. (114). 391—409.
- (Papatheodorou et al., 2002) C. Papatheodorou, A. Vassiliou, B. Simon: Discovery of Ontologies for Learning Resources Using Word-based Clustering, in: ED-MEDIA 2002. Copyright by AACE. Reprinted from the ED-MEDIA 2002 Proceedings, August 2002 with permission of AACE, Denver, USA, August, 2002.
- (Parrott et al., 2003) L. Parrott, R. Lacroix, and K.M. Wade. 2003. Design considerations for the implementation of multi-agent systems in the dairy industry. *Computers and Electronics in Agriculture*, 38(2) 79-98
- (Payne, et. al., 2002) Payne T., Singh R., and Sycara K. Communicating Agents in Open Multi-Agent Systems. *First GSFC/JPL Workshop on Radical Agent Concepts (WRAC)*, 2002.
- (Pechoucek, Marik, and Barta, 2001) Pechoucek M., Marik V., and Barta J. CplanT: An Acquaintance Model Based Coalition Formation Multi-Agent System. *Proceedings of the Second International Workshop of Central and Eastern Europe on Multi-Agent Systems (CEEMAS'2001)*, Krakow, Poland, 2001. 209—216.
- (PHP, 2003) PHP (2003). URL: <http://www.php.net>.
- (Puliafito et al., 2000) A. Puliafito, O. Tomarchio, and L. Vita. MAP: Design and Implementation of a Mobile Agents Platform. *Journal of System Architecture*, 46(2):145-162, January 2000. <http://sun195.iit.unict.it/Papers/jsa00.ps.gz>
- (Rathmell, 1999) Rathmell R.A. A Coalition Force Scenario “Binni – Gateway to the Golden Bowl of Africa. *Proceedings on the International Workshop on Knowledge-Based Planning for Coalition Forces* (ed. by A/ Tate). Edinburgh, Scotland, 1999. 115—125.
- (Roure, et. al., 2003) de Roure D., Jennings N. R., Shadbolt N. The Semantic Grid: A future e-Science infrastructure. *International Journal of Concurrency and Computation: Practice and Experience*, 2003. URL: <http://aspen.ucs.indiana.edu/CCPEwebresource/c590gridderoure/finalsource/c590semgrid-journal-v5.doc>.
- (Salis and Masili, 2002) Salis C. and Masili G. The User Control on Verbal/Non-verbal Knowledge Visualization. URL: <http://www.viu.unive.it/tedis/oslp/resources/papers/salis.htm>
- (Sandholm and Lesser, 1995) Sandholm T.W. and Lesser V.R. Issues in automated negotiation and electronic commerce: extending the contract net framework. *ICMAS-95 First International Conference on Multiagent Systems*, San Francisco, 1995.
- (Sandholm, 1993) Sandholm T. An Implementation of the Contract Net Protocol Based on Marginal Cost Calculations. *Proceedings of the National Conference on Artificial Intelligence*, Washington, D.C., 1993, July. 256—262.
- (Sandholm, 1999) Sandholm T. Distributed Rational Decision Making. *Multiagent Systems: A Modern Introduction to Distributed Artificial Intelligence* (G.Weiss, ed.). MIT Press, 1999. 201—258 URL: <http://www-2.cs.cmu.edu/~sandholm/rational.ps>
- (Shoham, 1993) Shoham Y. Agent-oriented programming. *Artificial Intelligence*, **60**, 1, 1993, 51—92.
- (Smirnov and Chandra, 2000) Smirnov A., Chandra C. Ontology-based Knowledge Management for Co-operative Supply Chain Configuration. *Proceedings of the 2000 AAAI Spring Symposium “Bringing Knowledge to Business Processes”*, March 20—22, Stanford, California, AAAI Press, 2000, 85—92.
- (Smirnov et al., 1997) Smirnov A., Pashkin M., Rakhmanova I. Multi-agent WWW-based Environment for Distributed Team Decision Making Support. *Proceedings of Benefit Day on Pan-European Co-operation and Technology Transfer*, Budapest, Hungary, 1997, 23—29.
- (Smirnov et al., 2002) Smirnov, A., Pashkin, M., Chilov, N. & Levashova, T. 2002. Business Knowledge Logistics: an Approach and Technology Framework. *Proceedings of the Sixth International Research Conference on Quality, Innovation & Knowledge Management (QIK'2002)*. Kuala Lumpur, Malaysia, February 17-20, 936-945.

- (Smirnov et al., 2003a) Smirnov A.V., Pashkin M.P., Chilov N.G., Levashova T.V. Ontology Management. *Journal of Computer and Systems Sciences International*, 2003, No. 4, 5
- (Smirnov et al., 2003b) Smirnov A., Pashkin M., Chilov N., Levashova T., Haritatos F. Knowledge Source Network Configuration Approach to Knowledge Logistics. *International Journal of General Systems*. Taylor & Francis Group, 2003, 32 (3), 251—269.
- (Smirnov et al., 2003c) Smirnov A., Pashkin M., Chilov N., Levashova T., Krizhanovsky A. In: R. Meersman, Z. Tari, D.C. Schmidt et al. (Eds.): *Ontology-Driven Knowledge Logistics Approach as Constraint Satisfaction Problem. On the Move to Meaningful Internet Systems 2003: CoopIS, DOA, and ODBASE*. – Proceedings of the International Conference on Ontologies, Databases and Applications of Semantics Information Reuse and Integration (ODBASE'2003). Catania, Sicily, Italy, 3—7 November 2003. Springer, 2003. V. 2888. 535—652.
- (Smirnov, 2000) Smirnov A. V. Rapid Knowledge Mass Customization Management Infosphere: Major Requirements and Technologies. Proceedings of 2000 Advanced Summer Institute (ASI 2000) and 2000 IIMB Workshop on Integration in Manufacturing and Beyond (IIMB 2000). Bordeaux, France, 2000. 356—358.
- (Smirnov, 2001a) Smirnov A. V. Rapid Knowledge Fusion into the Scalable Infosphere: A Concept and Possible Manufacturing Applications. *Proceedings of the International NAISO Congress on Information Science Innovations, Symposium on Intelligent Automated Manufacturing (IAM'2001)*. Dubai, U.A.E., 2001 (Electronic Proceedings).
- (Smirnov, 2001b) Smirnov A.V. Profile-Based Configuring of Knowledge Supply Networks in the Global Business Information Environment. Proceedings of the 2001 IEEE International Conference on Systems, Man, and Cybernetics (SMC'2001). Tuscon, Arizona, October 7—10, 2001, 977—982.
- (Smith, 1980) Smith R. The Contract Net Protocol: High-level Communication and Control in a Distributed Problem Solver. *IEEE Transactions on Computers*, 29 (12), 1980. 1104—1113.
- (SOAP, 2001) SOAP. MSDN Online Developer Center. Microsoft Corp., 2001. <http://msdn.microsoft.com/soap/>.
- (Sun, 2001) Java™ Remote Method Invocation (RMI). The Source for Java™ Technology. Sun Microsystems, Inc., 2001. <http://java.sun.com/products/jdk/rmi/>.
- (Sun, 2003) Sun corporate web-site, 2003. URL: <http://www.sun.com>.
- (The Globus Project Tutorial, 2003) The Globus Project. An Open Grid Services Architecture. Globus Tutorial, Argonne National Laboratory, January 29, 2002. URL: <http://www.globus.org/ogsa/deliverables/OGSA-January-2002-v3.pdf>
- (The Globus Project, 2003) The Globus Project, 2003. URL: <http://www.globus.org>.
- (Tomuro, 1998) Tomuro N. Semi-automatic Induction of Systematic Polysemy from WordNet. Proceedings of the workshop on Usage of WordNet in Natural Language Processing Systems at the 17th International Conference on Computational Linguistics (COLING-98), Montreal, Canada, 1998, 108—114.
- (UDDI.org, 2003) UDDI.org web site, 2003. URL: <http://www.uddi.org>.
- (UNSPSC, 2001) The UNSPSC Code (Universal Standard Products and Services Classification Code), DAML Ontology Library, Stanford University, January 2001. <http://www.ksl.stanford.edu/projects/DAML/UNSPSC.daml>
- (W3C, 2003) W3C Consortium, 2003. URL: <http://www.w3.org>.
- (Web3D, 2002) Web3D Consortium 2002. <http://www.vrml.org>.
- (Web-Onto, 2003) WebOnto: Knowledge Media Institute (KMI), The Open University, UK, 2003. <http://eldora.open.ac.uk:3000/webonto>.
- (Webopedia, 2001) Webopedia: The Only Online Dictionary and Search Engine you Need for Computer and Internet Technology. Internet.com Corp., 2001. <http://webopedia.internet.com>.
- (Wikipedia, 2003) Wikipedia, the free encyclopedia. <http://en.wikipedia.org>.
- (WonderWeb, 2002) WonderWeb: Ontology Infrastructure for the Semantic Web, 2002. <http://wonderweb.semanticweb.org>
- (Wong et al., 2003) Wong S.K., Nguyen T.T., Chang E., Jayaratna N. In: R. Meersman, Z. Tari, D.C. Schmidt et al. (Eds.): *Usability Metrics for E-learning. On the Move to Meaningful Internet Systems 2003: CoopIS, DOA, and ODBASE*. – Proceedings of the International Conference on Ontologies, Databases and Applications of Semantics Information Reuse and Integration. Catania, Sicily, Italy, 3-7 November 2003. Springer, 2003. V. 2888. 235—252.
- (WordNet, 2003) WordNet Web Site, 2003. URL: <http://www.cogsci.princeton.edu/~wn/>.
- (XML-RPC, 2003) XML-RPC Web-site, 2003. URL: <http://www.xmlrpc.org>.
- (XmlRpc++, 2003) XmlRpc++ Library Web-site, 2003. URL: <http://xmlrpcpp.sourceforge.net>.
- (Yokoo, 1999) Makoto Yokoo, Toru Ishida. Search Algorithms for Agents. In G. Weiss (ed.). *Multiagent Systems*. MIT Press, 1999.
- (Zadeh, 1994) Lotfi A. Zadeh: Fuzzy Logic, Neural Networks, and Soft Computing. *CACM* 37(3): 77-84 (1994).
- (Zeus, 2003) Zeus – BT Intelligent Agent Research. URL: <http://www.opensource.org>.
- (Zhang, et. al., 2002) Zhang H., Wu B., Dong M., Shi Z. Dynamic contract net protocol. *ICIT'2002 International Conference on Intelligent Information Technology*, Beijing, China, 2002. 564—572.